UNIJUI - UNIVERSIDADE REGIONAL DO NOROESTE DO ESTADO DO RIO GRANDE DO SUL

DCEEng – DEPARTAMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

ANÁLISE DE DESEMPENHO E CONSUMO ENERGÉTICO ENTRE PROCESSADORES ARM E x86

RICARDO KLEIN LORENZONI

Ijuí

DEZEMBRO/2011

UNIJUI - UNIVERSIDADE REGIONAL DO NOROESTE DO ESTADO DO RIO GRANDE DO SUL

DCEEng – DEPARTAMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

ANÁLISE DE DESEMPENHO E CONSUMO ENERGÉTICO ENTRE PROCESSADORES ARM E x86

RICARDO KLEIN LORENZONI

Trabalho de Conclusão de Curso apresentado ao Curso de Informática - Sistemas de Informação do Departamento de Ciências Exatas e Engenharias (DCEEng), da Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ), como requisito para a obtenção do título Bacharel em Informática - Sistemas de Informação.

Orientador: Prof. MSc. Edson Luiz Padoin

Ijuí

Dezembro/2011

ANÁLISE DE DESEMPENHO E CONSUMO ENERGÉTICO ENTRE PROCESSADORES ARM E x86

RICARDO KLEIN LORENZONI

Trabalho de Conclusão de Curso apresentado ao Curso de Informática - Sistemas de Informação do Departamento de Ciências Exatas e Engenharias (DCEEng), da Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ), como requisito para a obtenção do título Bacharel em Informática - Sistemas de Informação.

_____

Orientador: Prof. MSc. Edson Luiz Padoin

BANCA EXAMINADORA

_____

Prof. MSc. Rogério Samuel de Moura Martins

Ijuí

Dezembro/2011

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus por estar sempre presente em minha vida, por iluminar o meu caminho em todos os momentos, em todas as minhas conquistas, e me abençoar todos os dias com uma família maravilhosa.

A toda a minha família, principalmente meu pai, minha mãe e meus irmãos pelo apoio incondicional, esforço e escolhas que sempre fizeram em prol de minha educação e formação. Também, a minha namorada pela cumplicidade, amizade, companheirismo e amor sempre demonstrados.

A todos os professores que participaram da minha formação como pessoa, cidadão e profissional, pelas suas contribuições e conselhos dados dentro e fora da sala de aula, participando direta ou indiretamente da maioria das escolhas feitas por mim, durante todo esse período de formação e que continuarão a me influenciar durante o resto de minha vida.

Ao meu orientador por ter me guiado e me indicado os caminhos a seguir nos momentos de dúvida, durante a pesquisa e realização do trabalho.

Aos colegas, pela amizade, troca de ideias, e conhecimentos durante toda a formação acadêmica, onde todos batalharam arduamente em busca do sonho da graduação.

Ao GAIC (Grupo de Automação Industrial e de Controle) pela concessão de uso dos equipamentos que necessitei para a realização deste trabalho.

Também, ao bolsista, aluno e amigo Bruno Mokan Muenchen pelas contribuições ao meu trabalho, auxiliando nas pesquisas e na realização dos testes necessários para a conclusão do trabalho.

MUITO OBRIGADO!

RESUMO

Dentre as boas práticas da TI VERDE uma das mais abordadas na atualidade é o consumo eficiente de energia elétrica. Atualmente grande parte dos usuários domésticos tem por costume deixar seus computadores ligados várias horas por dia, muitas vezes sem executar nenhum programa, ou usando algum software de download. Grande parte das máquinas desses usuários são equipamentos mais antigos, ou máquinas destinadas a fornecer um grande desempenho, nestas máquinas, o consumo eficiente não é uma prioridade. A partir desse conhecimento, este trabalho visa medir o desempenho computacional e o consumo de energia de processadores da arquitetura ARM e de um processador mais antigo da arquitetura x86, com o intuito de avaliar a possibilidade de substituir estas máquinas subutilizadas, por máquinas mais econômicas e capazes de realizar o mesmo trabalho proporcionando uma grande economia de energia.

Palavras-chave: TI Verde; Desempenho de processadores; ARM; x86; MFLOPS; MFLOPS/WATT.

ABSTRACT

Among the best practices of GREEN IT one of the most discussed today is the efficient use of electricity. Currently most home users are customary to leave their computers powered on for several hours a day, often without running any programs, or by using any download software. Most of these machines are older equipment, or machines designed to deliver a great performance. In these machines, the energy efficiency is not a priority. From this knowledge, this study aims to measure the computational performance and power consumption of the ARM architecture processors and an older x86 architecture processor, in order to evaluate the possibility of replacing these machines underutilized by less expensive and capable machines performing the same job providing great energy savings.

Keywords: Green TI; Processor Performance; ARM; x86; MFLOPS; MFLOPS/WATT.

SUMÁRIO

1. INTRODUÇÃO 9
1.1. Problema 10
1.2. Hipóteses 11
1.3. Justificativa 11
1.4. Objetivo 13
1.4.1. Objetivo geral 13
1.4.2. Objetivos específicos 14
2. ESTADO DA ARTE 15
2.1. Arquiteturas x86 e ARM 15
2.1.1. Arquitetura x86 15
2.1.2. Arquitetura ARM 17
3. MENSURAÇÃO DO DESEMPENHO 19
3.1. MFLOPS (Milhões de Instruções de Ponto Flutuante por Segundo) 19
3.2. Benchmarks 19
3.2.1. Sintético 20
3.2.2. Kernel 21
3.2.3. Algoritmo 21
3.2.4. Aplicação 21
4. MENSURAÇÃO DO CONSUMO 23
4.1. Princípios da Eletricidade 23
4.1.1. Corrente 23
4.1.2. Tensão 25
4.1.3. Potência 26
4.1.4. Energia Consumida (consumo) 27
4.2. Equipamentos para medição 27

4.2.1. Dranetz Power Platform PP4300 27
4.2.2. Osciloscópio Agilent DSO1002A 28
4.2.3. Ponta de prova de tensão 29
4.2.4. Fonte para calibração da ponta de prova 29
5. ESTUDO DE CASO: 31
5.1. Descrição dos Equipamentos 33
5.1.1. A CPU 34
5.1.2. A BeagleBoard 34
5.2. Ambiente de Execução: 35
5.2.1. Sistema Operacional: 35
5.3. Linpack (HPL) 36
6. RESULTADOS 39
7. CONSIDERAÇÕES FINAIS 43
8. TRABALHOS FUTUROS 44
9. REFERÊNCIAS BIBLIOGRÁFICAS 45
ANEXOS 48

LISTA DE TABELAS

Tabela 1 - Especificações ARM e x86 35
Tabela 2 – Consumo médio na execução do algoritmo 39
Tabela 3 - Tempo médio de execução do algoritmo 39
Tabela 4 - % redução de consumo e tempo de execução utilizando otimizações na Arquitetura ARM 40
Tabela 5 - % redução de consumo e tempo de execução utilizando otimizações na Arquitetura x86 41
Tabela 6 - Desempenho / Watt 42

LISTA DE FIGURAS

Figura 1 - Plataforma BMI Dranetz PP4300 27
Figura 2 - Ponta de prova de corrente TR2510..... 28
Figura 3 - Osciloscópio Agilent DSO1002A 60MHz 28
Figura 4 - Ponta de prova de corrente 1146A 29
Figura 5 - Ponta de prova de osciloscópio LF-20A 29
Figura 6 - Fonte ICEL PS-1500 30
Figura 7 - Esquema de ligação dos equipamentos (ARM) 32
Figura 8 - Esquema de ligação dos equipamentos CPU 33
Figura 9 - Especificações de Energia BeagleBoard..... 34
Figura 10 - Fatoração da Matriz. 37
Figura 11 - Tela principal do IntuiLink Data Capture 3.90..... 51
Figura 12 - Tela de configuração para recuperar os dados do osciloscópio 52
Figura 13 - Ligando o aparelho. 54
Figura 14 – Inicialização do Dranetz. 54
Figura 15 - Selecionando a sondagem de corrente. 55
Figura 16 - Selecionando o tipo de sonda utilizada (TR2510). 55
Figura 17 – Tela principal do aparelho (acessando a tela de configurações). 56
Figura 18 - Acessando a opção 5: ADVANCED SETUP OPTIONS. 56
Figura 19 – Acessando a opção 1: SET INPUT CONFIGURATIONS 57
Figura 20 – Acessando a opção 3: SET SCALE FACTORS. 57
Figura 21 – Ajuste do fator de escala dos canais. 58
Figura 22 - Acessando a opção 4: SET SYNC PARAMETERS 58
Figura 23 – Ajustando a frequência da rede elétrica. 59
Figura 24 - Acessando a opção 5: SET INTERVAL FOR TIMED READINGS. 59
Figura 25 - Ajustando o intervalo em 5 segundos de medição dos dados. 60
Figura 26 - Ligando o aparelho. 60
Figura 27 - Inicialização do Dranetz. 61
Figura 28 – Confirmando a utilização de sondagem de corrente. 61
Figura 29 – Definindo o tipo da sonda utilizada. 62
Figura 30 – Acessando menu para desabilitar canais desnecessários. 62
Figura 31 – Canais B, C e D desativados. 63
Figura 32 – Alterando a visualização da medição de energia. 63
Figura 33 - Alterando para a próxima visualização da medição de energia. 64
Figura 34 – Tela utilizada para acompanhar as medições de energia. 64
Figura 35 – Acessando o menu de configurações. 65
Figura 36 - Acessando a opção 4: MEMORY FUNCTIONS. 65
Figura 37 - Acessando a opção 1: MEMORY CARD FUNCTIONS. 66
Figura 38 - Acessando a opção WRITE CARD para salvar os dados coletados. 66
Figura 39 - Salvar os dados coletados no cartão. 67

Figura 40 – Removendo o cartão de memória na frontal direita do aparelho..... 67
Figura 41 - Salvando os dados para o computador 68
Figura 42 - Tela principal do Dran-View..... 68
Figura 43 - Selecionando as informações desejadas. 69

1. INTRODUÇÃO

Nos últimos 30 anos, a discussão a respeito do esgotamento dos recursos naturais como os combustíveis fósseis, aumento da poluição e aumento exponencial de consumo de recursos energéticos tem aumentado a cada ano.

Na última década, essas discussões passaram a ganhar ainda mais força, contando não só com o apoio dos ecologistas, mas também com o apoio de profissionais de todas as áreas, bem como da população em geral. Na área de TI, o assunto é abordado com tentativas de reduzir a criação de lixo tecnológico causada pela rápida evolução dos dispositivos, e também com tentativas de reduzir o consumo de energia dos equipamentos.

Até alguns anos atrás, as indústrias competiam entre si em busca de criar processadores cada vez mais rápidos. Para vencer essa competição elas aumentavam a frequência de trabalho de seus produtos, até que chegaram aos limites que os processadores suportavam sem superaquecer e consequentemente derreter ou autodesligar.

Esse limite pode ser ampliado através do uso de coolers mais robustos e maiores com a finalidade única de resfriar o processador. Neste ponto as indústrias notaram que o custo de aumentar ainda mais a frequência de clock dos processadores iria se tornar alto demais, visto o grande desperdício de energia térmica nas máquinas. Além da grande quantidade de energia desperdiçada pelo processador em forma de energia térmica (calor), seria necessário utilizar coolers ainda maiores e mais robustos para manter essas máquinas resfriadas e funcionando.

Para solucionar esse problema, as indústrias optaram por reduzir a frequência de seus equipamentos (reduzindo a dissipação de energia térmica) e aumentar a sua capacidade de processamento. Fizeram isso aumentando o número de núcleos presentes dentro de cada processador, podendo assim, processar um maior número de informações, com uma mesma quantidade de energia, fazendo um consumo mais eficiente da mesma.

Com a grande difusão dos sistemas embarcados e da computação móvel, a microarquitetura ARM se tornou mais conhecida.

Os processadores ARM (Advanced RISC Machine) são desenvolvido pela ARM Holdings com o propósito de ter um baixo consumo de energia e obtendo o melhor desempenho possível com ela.

Os processadores dessa arquitetura seguem a risca algumas propostas básicas, são elas: ter um pequeno conjunto de instruções, ser pequenos (ocupar pouco espaço) e consumir pouca energia.

Com a crescente busca da população por produtos e serviços ecologicamente corretos ou com tecnologias limpas (não poluentes), as indústrias e organizações perceberam aí um grande nicho de mercado e passaram a investir nessa área, por esse investimento se tornar um grande diferencial competitivo para elas.

Existem no mercado alguns produtos na área da tecnologia que possuem processadores ARM que poderiam ser utilizados pelos consumidores de diversas formas, dentre elas, como computadores, pois os processadores ARM já possuem suporte oferecido por diversos sistemas operacionais.

Neste trabalho, pretende-se conhecer melhor a arquitetura x86 utilizada em desktops, notebooks, servidores, clusters e supercomputadores e a arquitetura ARM, presente na maioria dos celulares e sistemas embarcados do mundo. Após isso, pretende-se rodar nesses equipamentos alguns programas de benchmark, enquanto é mensurado o consumo de energia dos mesmos, para poder então compará-los com relação a consumo e desempenho.

1.1. Problema

Uma parcela dos usuários de sistemas computacionais tem o hábito de deixar os seus computadores ligados 24 horas por dia. Na grande maioria das vezes, esses equipamentos ficam ligados sem realizar nenhum tipo de tarefa, ou então estão realizando tarefas que não exigem grande capacidade de processamento, tais como realizar downloads ou atuar como um simples servidor de arquivos doméstico. Computadores mais antigos eram desenvolvidos pelos fabricantes, dando ênfase em desempenho, sem se preocupar com o consumo de energia gerado pelos mesmos. Um grande desafio hoje para a sociedade é realizar um consumo mais eficiente e consciente da energia elétrica, dessa forma, torna-se necessário, apresentar aos usuários e à sociedade, novas opções no que diz respeito a sistemas de computação com baixo consumo de energia e que supram as suas necessidades.

1.2. Hipóteses

Seriam os processadores ARM econômicos e com um desempenho computacional suficientemente elevado para serem utilizados como uma alternativa em dispositivos móveis e até mesmo em computadores, para substituir os processadores de usuários que necessitam de equipamentos com grande poder de processamento?

1.3. Justificativa

Em muitas empresas públicas ou privadas, grande parte dos computadores que são utilizados, possui processadores que fazem um alto consumo de energia, mesmo em momentos em que não estão sendo utilizados.

As ideologias de “ecologicamente correto” e “crescimento sustentável” se tornam cada vez mais presentes em nosso cotidiano, e vêm cada vez mais influenciando nas escolhas das pessoas entre um fornecedor ou outro, um fabricante ou outro, ou seja, cada vez mais, noticiar e fazer campanhas publicitárias divulgando que a marca/instituição faz um trabalho em busca de um “crescimento sustentável” ou “ecologicamente correto” vai se tornar um diferencial competitivo entre as empresas e organizações.

Um dos assuntos que mais cresce em todos os sentidos no mundo hoje é a economia de energia principalmente em notebooks, netbooks, celulares, tablets e PDA´s que utilizam baterias e necessitam de um controle mais rigoroso do consumo de energia e a utilização consciente dos recursos naturais.

Como o assunto não é mais somente o tempo de duração da carga, mas um consumo mais eficiente da energia disponível, evitando o desperdício, esse assunto deixa de estar somente voltado a dispositivos móveis e sistemas embarcados e passa a ser também uma necessidade a se avaliar em desktops, mainframes e servidores.

Para buscar o melhor desempenho considerando não só o tempo de resposta para uma solicitação, mas obter um tempo de resposta aceitável para a mesma fazendo um consumo eficiente da energia, as fabricantes de microprocessadores passaram a investir em pesquisas para melhorar a eficiência energética de seu

hardware, buscando afetar de forma mínima, ou até melhorar o desempenho de seus equipamentos.

A Intel nos últimos anos lançou os seus processadores, aumentando a quantidade de núcleos de processamento, reduzindo ou mantendo o clock para melhorar a eficiência dos mesmos sem aumentar a dissipação de calor. Após essa etapa, buscou melhorar a arquitetura interna de seus processadores, diminuindo o tamanho dos transistores e trocando o material dos mesmos que antes eram de Silício para Háfnio, um material menos tóxico, que conduz melhor a energia e que dissipa menos calor.

> Nos últimos modelos de processadores foram inúmeras as melhorias implementadas em nível de Hardware (redução do tamanho dos transistores, troca da matéria prima dos transistores) e Software (gerenciamento dinâmico dos cores, threads, cache e energia; adição de novas instruções SSE4; Multi-Threading simultâneo) sobre os seus processadores, na intenção de melhorar o desempenho do equipamento, consumir menos energia, e utilizá-la de forma mais eficiente (INTEL, 2011).

Com a constante evolução proporcionada pelas indústrias de microprocessadores, o desenvolvedor de software tem de se atualizar constantemente para ficar a par das especificações desses novos produtos que vêm chegando ao mercado, e assim deixar seus softwares melhor ajustados a estas tecnologias, aumentando ainda mais a eficácia das mudanças realizadas pelos fabricantes.

Com relação aos processadores com arquitetura ARM, vale ressaltar que os mesmos começaram a ser desenvolvidos em 1983, com seu primeiro modelo ficando pronto em 1985, porém não chegou ao mercado. Em 1986 a Arcon lançou no mercado o ARM2, desde lá, foram lançadas mais 9 versões do chip. Segundo Prado (2011) o diferencial dessa arquitetura com relação a arquitetura x86 é que ela segue alguns princípios como: ser um RISC de alta performance, ter o tamanho do núcleo reduzido, ter um baixo consumo de energia e excelente custo benefício.

Segundo Gomes, Leite & Caetano (2005) em 1986, o ARM2 era o processador de 32 bits mais simples, com apenas 30.000 transistores, mas ainda assim superava o desempenho de muitos processadores (RISC ou CISC) mais complexos, como o 286 da Intel.

> A ARM Ltd., que é a responsável pelo desenvolvimento dos chips e detentora dos direitos sobre a arquitetura, não produz os processadores, se limitando a licenciar os projetos a preços módicos para outros fabricantes, que podem optar por diversos tipos de licença, que vão de simples licenças para produzir os chips, a opções que permitem modificar os chips, incluir

> componentes adicionais, ou até mesmo ter acesso completo ao micro-código e desenvolver chips compatíveis (MORIMOTO, 2011).

Esta liberdade concedida aos fabricantes gera uma disputa saudável entre os mesmos por um melhor desempenho e consumo de energia, o que contribui para o avanço da tecnologia.

Conforme Maluta (2008) ao comparar a execução de um algoritmo de simples ordenação em processadores das duas arquiteturas, notou-se que a velocidade de execução nos processadores é proporcional a sua eficiência.

Portanto, a intenção deste trabalho é buscar o conhecimento acerca das arquiteturas, x86 e ARM, e a partir desse conhecimento selecionar um sistema completo (gabinete) que utilize um processador x86, e um sistema completo com processador ARM.

Após escolher esses equipamentos, será feito um conjunto de testes, com os dois , a fim de definir o consumo médio destes, em estado ocioso, e depois outros conjuntos de testes, para definir o desempenho de processamento dos dois equipamentos, juntamente com o consumo de energia realizado pelos mesmos durante a execução desses testes, para então poder definir se um processador da arquitetura ARM pode substituir processadores x86 em computadores de uso pessoal. .

1.4. Objetivo

1.4.1. Objetivo geral

- Analisar o consumo de energia de dispositivos e correlacionar o consumo de energia com a quantificação de seu desempenho, e apresentar os resultados, indicando qual tem um melhor desempenho/Watt consumido.

1.4.2. Objetivos específicos

- Medir o consumo de energia de dispositivos das arquiteturas x86 e ARM;
- Quantificar o desempenho dos processadores por meio de benchmarks;

2. ESTADO DA ARTE

Inicialmente pretende-se abordar um pouco as arquiteturas x86 e ARM, apresentando algumas diferenças entre elas, em seguida apresentar a métrica de medida de desempenho de computadores MFLOPS, e ao final, definir o conceito de benchmark e apresentar alguns dos diferentes tipos de classificação de benchmark, definindo o conceito de cada um.

2.1. Arquiteturas x86 e ARM

Este subcapítulo tem por objetivo apresentar as características principais das arquiteturas x86 e ARM, com relação ao conjunto de instruções e características próprias das mesmas.

Uma das principais particularidades de ambas as arquiteturas, é que são superes calares.

> Um processador superes calar é aquele no qual são usadas várias pipelines de instrução independentes. Cada pipeline tem diversos estágios, podendo manipular várias instruções a cada instante. O uso de várias pipelines introduz um novo nível de paralelismo, possibilitando processar diversos fluxos de instrução de cada vez. Um processador superes calar explora o que é conhecido como paralelismo no nível de instruções, que diz respeito ao nível em que instruções de um programa podem ser executadas em paralelo (STALLINGS, 2002).

2.1.1. Arquitetura x86

Segundo Torres e Lima (2006), os processadores da arquitetura x86, utilizam uma arquitetura híbrida CISC/RISC. A solução implantada hoje por todos os processadores CISC é o uso de um decodificador CISC/RISC, pois o processador trabalha internamente executando instruções RISC, mas aceita apenas instruções CISC que no processador são decodificadas e então processadas em formato RISC.

> O design de um conjunto de instruções de um computador deve levar em consideração não apenas a construção de uma linguagem de máquina, mas também os requerimentos impostos no uso de programação com linguagens de alto nível. A tradução de linguagens de alto nível para linguagem de máquina é feita por compiladores. Uma razão para que a tendência seja de um conjunto de instruções complexo é o desejo de simplificar a compilação e aumentar a performance como um todo. A tarefa de um compilador é gerar uma sequência de instruções de máquina para cada conjunto de códigos de alto nível. A tarefa é simplificada se houverem instruções de máquina que implementam diretamente os conjuntos de códigos de alto nível. O objetivo principal de uma arquitetura CISC é tentar

> prover uma única instrução de máquina para cada código escrito em uma linguagem de alto nível (MANO, 1993).

Segundo Mano (1993) as principais características dos processadores CISC são:

- Um grande número de instruções – tipicamente de 100 a 250 instruções;
- Algumas instruções que desempenham tarefas específicas e não são usadas com tanta frequência;
- Uma grande variedade de modos de endereçamento;
- Instruções com comprimento variável;
- Instruções que manipulam operandos na memória.

Possuem um conjunto de instruções com tamanho variável:

> Codificações de conjuntos de instruções de comprimento variável utilizam diferentes números de bits para codificar as instruções, dependendo da quantidade de entradas da instrução, dos modos de endereçamento utilizados e de outros fatores. Ao utilizar a codificação de comprimento variável, cada instrução toma apenas o espaço de memória que é necessário, se bem que muitos sistemas exijam que todas as codificações de instruções tenham um número inteiro de bytes de comprimento. Utilizar um conjunto de instruções de comprimento variável pode reduzir a quantidade de espaço ocupado por um programa, mas aumenta enormemente a complexidade da logica necessária para decodificar as instruções, uma vez que partes da instrução, como operandos de entrada, podem ser armazenadas em diferentes posições de bit, em diferentes instruções. Além disto, o hardware não pode predizer a localização da próxima instrução até que a instrução atual tenha sido decodificada o suficiente para que ele possa saber o comprimento da instrução atual. (CARTER, 2003).

Conforme Murdocca e Heuring (2000) É importante notar que os arquitetos de computadores aprenderam que instruções CISC não funcionam muito bem em arquiteturas com pipeline. Para que pipeline funcione de forma efetiva, cada instrução necessita ter similaridades com outras instruções, pelo menos em termos da complexidade relativa das instruções.

Desta forma, percebe-se o motivo de hoje os processadores da arquitetura x86 serem híbridos CISC/RISC, onde o processador recebe instruções CISC e as converte em instruções RISC, buscando otimizar o processamento através do uso de pipeline.

2.1.2. Arquitetura ARM

Conforme Murdocca e Heuring (2000) a arquitetura RISC tem algumas características que a diferem da arquitetura CISC:

> - Todas as instruções têm tamanho fixo;
> - Todas as instruções efetuam operações simples que podem ser iniciadas no pipeline à taxa de uma por ciclo de clock. Operações complexas são agora construídas a partir de instruções simples pelo compilador;
> - Todos os operandos devem estar em registradores antes de serem modificados. Existe uma classe distinta para instruções de acesso a memória: LOAD e STORE. Esta é chamada de arquitetura LOAD-STORE;
> - Modos de endereçamento são limitados. Cálculos complexos de endereços são feitos usando-se de sequências de operações simples;
> - Deve existir um número grande de registradores de uso geral para operações aritméticas para que variáveis temporárias possam ser armazenadas em registradores em vez de em uma pilha na memória (MURDOCCA e HEURING, 2000).

Segundo Gomes, Leite e Caetano, as principais características da arquitetura ARM são as seguintes:

> • Processador de 32 bits;
> • 16 registradores de uso geral;
> • Conjunto de instruções extensível com o uso de coprocessadores;
> • Instruções básicas similares ao 6502;
> • Instruções de três endereços;
> • Capacidade de executar instruções de 16 bits usando
> a arquitetura Thumb;
> • Baixo consumo de energia;
> • Tamanho do núcleo reduzido;
> • Até 16 coprocessadores lógicos. (GOMES, LEITE e CAETANO, 2011)

> A arquitetura ARM foi desenvolvida para possibilitar implementações muito enxutas, sem deixar de lado o alto desempenho. Isso é possível pela simplicidade dos processadores ARM. Importante lembrar que implementações pequenas implicam em baixo consumo de energia, o que torna esses processadores interessantes para aplicações móveis. (GOMES, LEITE e CAETANO, 2011).

A tecnologia ARM, que iniciou seu desenvolvimento em 1978, vem crescendo cada vez mais no mercado nos últimos 10 anos, tendo em vista que não possui nenhum concorrente direto no mercado, possui uma grande fatia do mesmo na área de processadores embarcados, principalmente no ramo de SmartPhones e Tablets. A Arquitetura ARM vem se tornando cada vez mais uma tendência e pode vir a se tornar um grande concorrente aos processadores da Intel, AMD entre outras, com a atual proposta da Microsoft, desenvolvedora do Windows, que pretende tornar seu

principal produto, compatível com essa tecnologia a partir da próxima versão do sistema operacional.

> O ARM é tipicamente um RISC (Reduced Instruction Set Computer). Algumas características interessantes das instruções ARM:
> - Conjunto grande e uniforme de registradores;
> - Arquitetura de LOAD / STORE. Operações de processamento de dados não operam diretamente com o conteúdo da memória, somente com o conteúdo de registradores;
> - Modos de endereçamento simples, com todos endereços de load / store sendo determinados a partir dos registradores ou pelos campos da instrução;
> - Uniformidade e tamanho fixo dos campos das instruções para simplificar a decodificação de instruções;
> - Controle sobre a ALU e sobre o shifter (deslocador) em todas instruções de processamento de dados;
> - Auto incremento e decremento dos endereços das instruções;
> - Instruções de múltiplos loads / stores para maximizar a performance;
> - Execução condicional da maioria das instruções (GOMES, LEITE e CAETANO, 2011).

Possui conjunto de instruções de tamanho fixo e segundo Carter (2003):

> Codificações de conjuntos de intruções de tamanho fixo utilizam o mesmo número de bits para codificar cada instrução. Codificações de comprimento fixo têm a vantagem por serem simples de decodificar, reduzindo a quantidade de lógica e latência da lógica de decodificação. Além disso, um processador que utilize codificação de comprimento fixo para o seu conjunto de instruções pode facilmente prever a localização da próxima instrução a ser executada (assumindo que a instrução atual não é um desvio). Isto faz com que seja mais fácil para o processador utilizar pipelining para melhorar o desempenho ao sobrepor a execução de várias instruções (CARTER, 2003).

3. MENSURAÇÃO DO DESEMPENHO

Neste capítulo será apresentado um pouco a respeito dos motivos para os quais se pode medir o desempenho assim como algumas das métricas definidas para se quantificar o desempenho.

Utilizar métricas para demonstrar o desempenho se faz necessário, para que se possam comparar dois equipamentos distintos de forma imparcial.

3.1. MFLOPS (Milhões de Instruções de Ponto Flutuante por Segundo)

Conforme Silva e Prado (2010) As unidades mais comuns de medida de desempenho em processamento de alto desempenho são a MFLOPS (Millions of floating-point operations per second) ou Megaflops.

Segundo Tsukamoto (2010) a velocidade de processamento de um supercomputador é medida em FLOPS (Floating Point Operations Per Second), ou seja, é quantidade de operações de ponto flutuante por segundo.

Ainda segundo Tsukamoto (2010) os supercomputadores atuais atingem velocidades na grandeza dos petaflops (PFLOPS), o que equivale a quatrilhões de cálculos por segundo.

> FLOPS (ou flops) é acrônimo de computação que significa Floating point Operations Per Second, que, em português, quer dizer operações de ponto flutuante por segundo. Isto é usado para determinar o desempenho de um computador, especificamente no campo de cálculos científicos.
> Já que dispositivos de computação têm enorme capacidade de processamento, convém utilizar unidades maiores que FLOPS, seus múltiplos. Os múltiplos mais utilizados são: megaflops (MFLOPS), gigaflops (GFLOPS), teraflops (TFLOPS), petaflops (PFLOPS) e exaflops (EFLOPS) (FLOPS, 2011).

3.2. Benchmarks

> Benchmarks são programas utilizados para avaliar sistematicamente algum aspecto do desempenho de um computador, com o objetivo de encontrar gargalos no sistema, ou para comparar o desempenho de duas ou mais máquinas diferentes. Quando utilizado para comparar o desempenho de máquinas, a métrica de desempenho mais confiável é o tempo de execução total para executar o benchmark (LILJA, 2000).

> Um benchmark é um programa de teste de performance que visa adquirir características de processamento e movimentação de dados de um grupo

> de aplicações. Programas de benchmark são usados para medir e prever a performance de um sistema e revelar seu comportamento bem como revelar seus pontos fortes e fracos. Ainda, definimos uma suíte de benchmark como um grupo de programas de benchmark juntamente com um conjunto de regras para governar condições e procedimentos de teste, incluindo a plataforma do ambiente, entrada e saída de dados e medidas de performance (OBAIDAT e BOUDRIGA, 2010).

Conforme Ida (2000) Benchmarks são programas que têm por finalidade medir a performance dos componentes de sistemas de computação, a necessidade de fazer uso destes programas é para comparar diferentes sistemas computacionais, a fim de verificar a melhor relação custo/beneficio. Além disso, é extremamente importante nos testes de eficiência entre hardware/compilador. Além de ser uma ótima ferramenta para detecção e análise de gargalos no sistema. Há uma grande discussão em torno de concretizar uma padronização dos benchmarks e de suas respectivas unidades de medida, porém esta é uma tarefa difícil, dadas as variadas arquiteturas de computadores disponíveis e a sua velocidade de evolução.

Conforme Ida (2000) Apud Berry, Cybenko e Larson (1991) dependendo da codificação do programa, os benchmarks podem ser divididos em vários tipos, dentre os principais estão:

3.2.1. Sintético

Conforme Ramos (2008) um Benchmark sintético é um programa artificial no qual o conjunto de operações que ele executa é selecionado para simular uma determinada classe de aplicações. O objetivo é que sendo o conjunto de instruções, o mesmo, das aplicações reais, o desempenho obtido na execução será um bom parâmetro para a avaliação do desempenho da aplicação real.

> Benchmarks sintéticos são pequenos programas artificiais que contem uma mistura de afirmações selecionadas de modo que eles representem um grande conjunto de aplicações reais. Benchmarks sintéticos normalmente não executam operações significativas em um grande conjunto de dados. Eles assumem o risco de que algumas partes dos programas podem ser removidas por um compilador de otimização (RAUBER e RÜNGER, 2010).

> São aqueles cujo código não faz nenhuma computação útil, não representa nenhuma aplicação real; somente exercita alguns componentes básicos do computador. Geralmente, tentam determinar uma frequência média de instruções típicas, comumente utilizadas, e recriá-las em um programa. Os mais conhecidos são o Whetstone e o Dhrystone (IDA, 2000).

> A maioria dos aplicativos projetados atualmente não só utilizam o processamento, mas também necessita de uma grande quantidade de operações de input / output com a ajuda de dispositivos externos. Assim, estes se tornaram uma parte integrante da carga de trabalho, que precisa ser considerada para a avaliação do desempenho do sistema no mundo real (OBAIDAT e BOUDRIGA, 2010)

3.2.2. Kernel

> São baseados no fato de que a maior parte da computação de um programa é concentrada em uma pequena parte de seu código. Esta pequena parte, chamada de núcleo (kernel), é extraída do programa e usada como benchmark. Deve ser ressaltado que eles não servem para avaliar completamente o desempenho de uma máquina. São bastante interessantes por sua simplicidade e pequeno tamanho. Um bom exemplo para este tipo de programa é o Livermore Loops (IDA, 2000).

> Conforme Thomas (2010) benchmarks kernel são partes pequenas mas relevantes de aplicações reais que normalmente capturam uma grande parcela do tempo de execução de aplicações reais. Comparado a programas reais, benchmarks kernel têm a vantagem de que são muito mais curtos e fáceis de analisar (RAUBER e RÜNGER, 2010).

> Benchmarks de kernel não consideram os dispositivos de entrada e saída, caracterizando o desempenho dos sistemas apenas com base na performance do processador. A grande desvantagem de utilizar estes tipos de benchmark é que eles não fazem uso de dispositivos de I/O para o cálculo do desempenho do sistema, assim, o desempenho do kernel não irá visualizar o desempenho do sistema como um todo (OBAIDAT e BOUDRIGA, 2010).

3.2.3. Algoritmo

Para Ida (2000) são algoritmos bem definidos, geralmente implementações de métodos conhecidos em computação numérica, como por exemplo os métodos de resolução de equações lineares (álgebra linear) que fazem parte do benchmark Linpack.

3.2.4. Aplicação

Segundo Ida (2000) benchmarks de aplicação são programas completos, que resolvem problemas científicos bem definidos. Um exemplo é o SPEC.

> Benchmarks do tipo aplicação são utilizados principalmente para avaliar o desempenho dos sistemas, que são usadas para uma determinada aplicação, tais como reserva de linha aérea, serviços bancários, classificação, previsão do tempo, e assim por diante. Para essas

aplicações, benchmarks são definidos como um grupo coletivo de funções, que fazem uso de todos os recursos do sistema (OBAIDAT e BOUDRIGA, 2010).

Benchmarks de aplicativos reais incluem vários programas inteiros que refletem uma carga de trabalho de um usuário padrão. Tais coleções são frequentemente chamadas de suítes de referência. Eles têm a vantagem de que todos os aspectos dos programas selecionados são capturados. Os resultados de desempenho produzidos são significativos para os usuários para os quais o pacote de benchmark representa as cargas de trabalho típico (RAUBER e RÜNGER, 2010).

4. MENSURAÇÃO DO CONSUMO

Neste capítulo serão apresentados alguns conceitos básicos de eletricidade como tensão, corrente, potência e consumo, e em seguida, alguns equipamentos que podem ser utilizados para medir essas grandezas.

4.1. Princípios da Eletricidade

Nesta subseção, serão definidos alguns dos princípios da eletricidade, como a corrente alternada, a corrente contínua, Tensão e Potência, para depois, com o conhecimento teórico a respeito desses tópicos, discutir a respeito dos mesmos no desenvolvimento do trabalho.

4.1.1. Corrente

Corrente elétrica é o movimento de partículas carregadas numa direção especifica. Esta partícula carregada poder ser tanto um elétron, quanto um íon positivo ou um íon negativo. Esta partícula é denominada com portador de corrente. O movimento pode ser através de um solido, um gás, um liquido ou o vácuo. (FOWLER, 1992; GUSSOW, 1996; O'MALLEY, 1993)

Segundo Fowler (1992) a unidade base de corrente é denominada ampère, que é igual a 1 coulomb por segundo.

> A corrente elétrica é o resultado do movimento de cargas elétricas. A unidade SI de corrente é o ampère, cujo símbolo é A. os símbolos utilizados são o I para uma corrente constante e i para uma corrente variável no tempo. Se um fluxo constante de 1C de carga passa por um dado ponto em um condutor durante 1 s, a corrente resultante é 1 A (O'MALLEY, 1993).

> O movimento ou o fluxo de elétrons é chamado corrente. Para se produzir corrente, os elétrons devem se deslocar pelo efeito de uma diferença de potencial. A corrente é representada pela letra I. A unidade fundamental com que se mede a corrente é o ampère (A). Um ampère de corrente é definido como o deslocamento de um coulomb através de um ponto qualquer de um condutor durante um intervalo de tempo de um segundo (GUSSOW, 1996).

Segundo Fowler (1992) a abreviatura para ampère é A. Por exemplo, para indicar que a corrente num fio é de 10 ampères, escrevemos I = 10 A.

$$I = \frac{\mathrm{Q}}{\mathrm{T}}$$

Onde *I* = corrente, A

Q = carga, C (coulomb)

T = tempo, s

CORRENTE ALTERNADA:

> A corrente alternada se comporta da maneira sugerida pelo próprio nome. Os elétrons se movem no circuito primeiro em um sentido, depois no sentido oposto, oscilando para cá e para lá em torno de posições fixas. Isso é realizado por uma alternância de polaridade da voltagem do gerador ou de outra fonte de voltagem. Aproximadamente todos os circuitos de CA comerciais na América do Norte envolvem voltagens e correntes que se alternam de um lado para o outro com uma frequência de 60 ciclos por segundo. Isso dá origem a uma corrente de 60 Hertz (HEWITT, 2006).

> Uma fonte de tensão alternada (Tensão ca) inverte ou alterna periodicamente a sua polaridade. Consequentemente, o sentido da corrente alternada resultante também é invertido periodicamente. Em termos do fluxo convencional, a corrente flui do terminal positivo da fonte de tensão, percorre o circuito e volta para o terminal negativo, mas quando o gerador alterna a sua polaridade, a corrente tem de inverter o seu sentido (GUSSOW, 1996).

Segundo Roditi (2005) é a corrente elétrica que possui variação periódica, em geral senoidal, com o tempo; sua magnitude é medida no seu valor máximo ou no seu valor quadrático médio. Na corrente alternada, a cada ciclo ocorre uma inversão no sentido da corrente. [símbolo ac, ca, AC ou CA]

CORRENTE CONTINUA:

> Corrente contínua se refere ao fluxo de cargas em um único sentido. Uma bateria produz uma corrente contínua em um circuito porque seus terminais sempre possuem a mesma polaridade. Os elétrons se movem do terminal negativo, que os repele, para o terminal positivo, que os atrai, sempre no mesmo sentido de movimento ao longo do circuito (HEWITT, 2006).

> A corrente contínua passa através de um condutor ou de um circuito somente num sentido. A razão dessa corrente unidirecional se deve ao fato das fontes de tensão, como as pilhas e as baterias, manterem a mesma polaridade de tensão de saída. A tensão fornecida por essas fontes é chamada de tensão de corrente contínua ou simplesmente de tensão dc ou tensão cc. Uma fonte de tensão contínua pode variar o valor da usa tensão de saída, mas se a polaridade for mantida, a corrente fluirá somente num sentido (GUSSOW, 1996).

Para Roditi (2005) é a corrente elétrica que flui numa única direção e com magnitude praticamente constante. [símbolo cc ou DC].

4.1.2. Tensão

> A tensão é a pressão elétrica que causa a circulação de corrente. A tensão é também conhecida como força eletromotriz (fem), ou diferença de potencial. Todos estes termos se referem à força que coloca cargas em movimento. A diferença de potencial é o termo que melhor descreve o fenômeno porque uma tensão é uma diferença de energia potencial que existe entre dois pontos. O símbolo da tensão é V(volt) (FOWLER, 1992).

Segundo Fowler (1992) a unidade base de tensão é o joule por coulomb. Um joule por coulomb é igual a um volt. O volt é a unidade base de tensão. A abreviatura é V.

De acordo com Gussow (1996) a unidade fundamental de diferença de potencial é o volt (V). O símbolo usado para a diferença de potencial é V, que indica a capacidade de realizar trabalho ao se forçar os elétrons a se deslocarem. A diferença de potencial é chamada tensão.

> A força que ocasiona o movimento de elétrons livres em um condutor, formando uma corrente elétrica, é chamada (1) força eletromotriz (fem), (2) tensão ou (3) diferença de potencial. Quando existe uma diferença de potencial entre dois corpos carregados que são ligados por um condutor, os elétrons fluirão ao longo do condutor. Esse fluxo de elétrons se dará do corpo carregado negativamente para o corpo carregado positivamente, até que as duas cargas sejam igualadas e que não mais exista diferença de potencia (MARINHA DOS ESTADOS UNIDOS, 1998).

> O conceito de tensão envolve trabalho, que por sua vez envolve força e distância. A unidade SI de trabalho é o joule, cujo símbolo é J. A unidade de força é newton, sujo símbolo é N e, conforme citado, a unidade SI de distância é o metro, cujo símbolo é m.
> Para se mover um objeto é necessário trabalho, para que seja vencida a força existente em oposição ao movimento. Por exemplo, levantar alguma coisas opondo-se à força da gravidade requer trabalho. De forma genérica, o trabalho requerido, em joules, é o produto da força em newton pela distância de deslocamento em metros:
>
> $$\mathrm{W\ (joules)} = \mathrm{F(newtons)X\ s(metros)}$$
>
> Onde W, F e s são símbolos de trabalho, força e distância respectivamente.
> Energia é a capacidade de realizar trabalho. Uma de suas formas é a energia potencial, que é a energia que um corpo possui devido a sua posição (O'MALLEY, 1993).

O'malley (1993) diz que a diferença de tensão entre dois pontos, também chamada de diferença de potencial, é o trabalho necessário em joules para mover 1C de carga de um ponto a outro. A unidade SI de tensão é o volt, cujo símbolo é V. O símbolo de tensão é V ou v, sendo utilizados também E ou e. Assim,

$$V\ (volts) = \frac{W\ (joules)}{Q\ (coulombs)}$$

4.1.3. Potência

Para Fowler (1992) a potência se refere a quão rapidamente a energia é usada ou convertida noutra forma de energia. Desde que a energia é a capacidade em realizar trabalho, pode-se dizer que a potência está relacionada com a rapidez na realização do trabalho.

Segundo Fowler (1992) combina-se estas duas ideias na definição de potência, que é a taxa de uso da energia ou realização de trabalho. O símbolo de potência é P.

De acordo com Fowler (1992) na eletricidade, o joule é usado como unidade base de energia e o segundo como unidade base de tempo. Portanto a unidade de potência é o joule por segundo (J/s). Ele é denominado watt em homenagem ao físico e inventor escocês James Watt.

Para Fowler (1992) a unidade base de potência é o watt, que é igual a 1 J/s e cuja abreviatura é W.

$$Potência\ (P) = \frac{\text{Energia (W)}}{\text{Tempo (t)}}$$

> A unidade básica de potência é o WATT e é igual a tensão aplicada ao circuito multiplicada pela corrente que flui neste circuito. Isso representa a razão, em qualquer instante, de execução de trabalho pelo movimento de elétrons no circuito. O símbolo P indica potência elétrica. Assim a fórmula de potência é: P=EI. E é a tensão e I é a corrente que flui no resistor ou no circuito cuja potência está sendo medida. A quantidade de potência mudará quando a tensão, a corrente, ou ambos, mudarem (MARINHA DO ESTADOS UNIDOS, 1998).

> A razão na qual algum corpo absorve ou produz energia é a potência absorvida ou produzida por este corpo. Uma fonte de energia produz ou desenvolve energia, e uma carga absorve energia. A unidade SI da potência é o watt, cujo símbolo é W. O símbolo de potência é P para potências constantes e p para potências variáveis no tempo. Se 1 J de trabalho é absorvido ou liberado numa taxa constante de 1 s, a potência correspondente é 1W (O'MALLEY, 1993).

CALCULO DE POTÊNCIA:

> A potência é igual à corrente vezes tensão. Expresso como uma fórmula, tem-se:
>
> Potência (P) = corrente (I) X tensão (V)
>
> Ou
>
> P = VI
>
> A potência está na sua unidade base quando a tensão é dada em volts e a corrente em ampères (FOWLER, 1992).

4.1.4. Energia Consumida (consumo)

A energia elétrica consumida ou produzida é o produto da corrente elétrica de entrada ou saída e o tempo durante o qual essa entrada ou saída ocorre:

$$\mathrm{W(joules)} = \mathrm{P(watts)X\ t(segundos)}$$

Energia elétrica é aquilo que os consumidores compram das companhias elétricas. Essas companhias não usam o joule como unidade de energia. A unidade utilizada é o quilowatt-hora (KWh), por ser maior e mais conveniente. O KWh não é uma unidade SI. O número de KWh consumidos é igual ao produto da potência absorvida em KW e o tempo durante o qual ocorreu esse consumo:

$$\mathrm{W\ (quilowatts - hora)} = \mathrm{P(quilowatts)X\ t(horas)}$$

(O'MALLEY, 1993).

4.2. Equipamentos para medição

Neste subcapítulo sugerem-se alguns equipamentos que podem ser utilizados para medir as grandezas citadas no subcapítulo anterior. O Objetivo disso é adquirir um conhecimento mínimo a respeito de equipamentos utilizados na medição de grandezas elétricas, e fazer uso de alguns deles para a realização do trabalho.

4.2.1. Dranetz Power Platform PP4300

O Dranetz Power Platform 4300 é um equipamento de alta precisão, que fornece inúmeros tipos de dados ao usuário, como a Tensão, Corrente, Potência, Consumo, e Consumo instantâneo. Suas medidas podem ser extraídas através de memória flash e analisadas por meio do software Dran-View, uma ferramenta do software proprietária do fabricante do equipamento.

Figura 1 - Plataforma BMI Dranetz PP4300

4.2.1.1. Ponta de prova de corrente TR2510

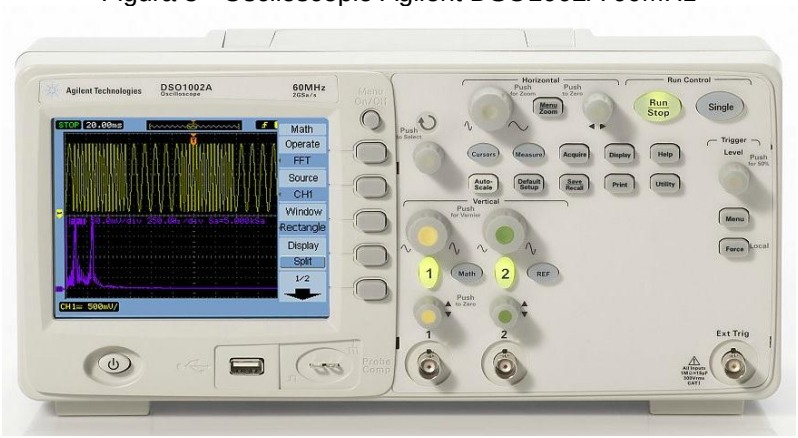
A ponta de prova de corrente TR2510 é uma das pontas de prova de corrente específicas para ser utilizada no Dranetz. Simples e fácil de usar, para que comece a informar os dados coletados por ela, basta conectar no aparelho e ligá-lo. Porém, é interessante que se verifique no Dranetz se a escala está travada em 1A.

Figura 2 - Ponta de prova de corrente TR2510

4.2.2. Osciloscópio Agilent DSO1002A

Segundo Roque (2007) o osciloscópio é um instrumento de medida destinado a visualizar um sinal eléctrico. Como muitas grandezas físicas são medidas através de um sinal eléctrico, e que o osciloscópio é um instrumento muito sensível à tensão, permite obter os valores instantâneos de sinais eléctricos rápidos, a medição de tensões e correntes eléctricas, e ainda frequências e diferenças de fase de oscilações.

Figura 3 - Osciloscópio Agilent DSO1002A 60MHz



4.2.2.1. Ponta de prova de Corrente 1146A

A ponta de prova de corrente 1146A é uma ferramenta específica para uso em osciloscópios fabricada pela Agilent, fácil de configurar e calibrar, é altamente confiável e precisa.

Figura 4 - Ponta de prova de corrente 1146A

4.2.3. Ponta de prova de tensão

A ponta de prova LF-20A é fabricada para uso exclusivo em osciloscópios para a mensuração de tensão.

Figura 5 - Ponta de prova de osciloscópio LF-20A

4.2.4. Fonte para calibração da ponta de prova

A fonte ICEL OS-1500 foi utilizada para calibrar a ponta de prova de corrente 1146A do Osciloscópio, fazendo com que a precisão da medida fornecida pela ponta de prova, fosse próxima dos 100%.

Figura 6 - Fonte ICEL PS-1500

5. ESTUDO DE CASO:

O objetivo deste trabalho é avaliar o desempenho computacional de um processador presente atualmente nos celulares (ARM), e de um processador de desktop (x86). Para tais avaliações, optou-se por rodar como benchmarks, um programa que executa a multiplicação de duas matrizes de ordem 1000 e o HPL (High Performance Linpack).

Este programa foi executado em modo sequencial e com 2(duas) threads, compilado nos modos default, com otimização –O1 e –O2. O desempenho foi avaliado através do tempo de execução dos algoritmos obtidos através da biblioteca tempo.h presente na sessão 4.4, e através da execução do benchmark HPL, parte integrante do Linpack.

A execução do HPL retorna ao usuário um valor em MFLOPS ou GFLOPS, que representa o número de operações de ponto flutuante executadas num intervalo de 1 segundo pelo processador.

Além do desempenho computacional, buscou-se também, mensurar o consumo de energia, realizado pelos equipamentos durante a execução destes algoritmos, para avaliar o desempenho deles com relação ao consumo de energia.

Outra forma de medir o desempenho de equipamentos é através do tempo de execução, esta metodologia também foi utilizada na realização deste trabalho. Pois o tempo de execução de alguma tarefa também é um fator crucial para o usuário dependendo de qual a pretensão de uso que o mesmo pretende fazer de um equipamento.

Os equipamentos avaliados representam duas classes distintas de hardwares, sendo um deles, um desktop padrão, com processador Intel Pentium IV denominado como x86 que tem como prioridade o alto desempenho. O outro é uma placa Beagleboard rev. C3 da Texas Instrumens que simula um dispositivo móvel, com a maioria das tecnologias já embutidas, entre elas o processador, memória RAM, portas para periféricos USB, USB OTG, SD, uma porta DVI-D, uma porta S-Vídeo, uma porta de saída de áudio stereo, uma entrada de áudio, uma porta serial e a entrada da alimentação. Esta placa simula um computador completo, com o benefício de ser pequena, consumir pouca energia e não fazer nenhum barulho.

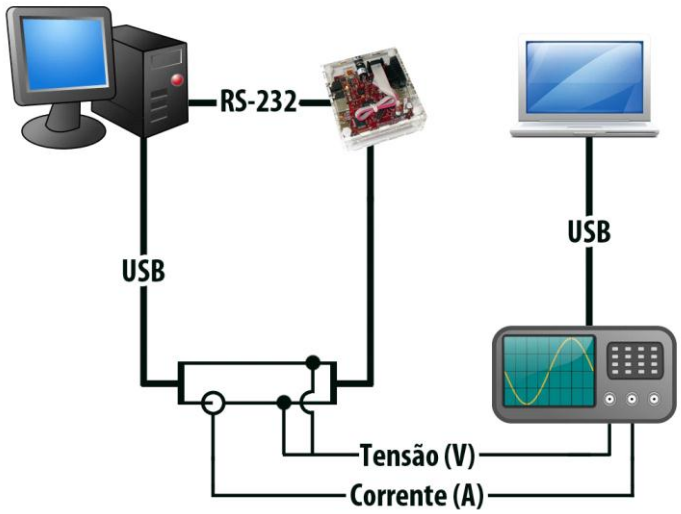
O primeiro conjunto de testes foi realizado utilizando uma BeagleBoard, que é um sistema desenvolvido pela Texas Instruments com o objetivo de demonstrar o

uso do processador OMAP-3530 (arquitetura ARM). Inicialmente foi realizada uma medição deste sistema em estado ocioso, para obter um valor médio referente ao seu consumo. Posteriormente foram executados os testes para medir o desempenho do equipamento e o consumo de energia do equipamento durante a execução dos mesmos.

A execução do benchmark HPL, ocorreu de forma diferenciada, onde primeiramente foram realizadas diversas execuções diferentes do algoritmo no equipamento para obter-se o melhor resultado possível, ao encontrar tal resultado, foi realizada mais uma execução de tal teste, enquanto se media o consumo do equipamento.

Para as medições de consumo com a Beagleboard foi usado um Osciloscópio Agilent Technologies DSO1002A, equipado com uma ponta de prova Agilent Technologies 1146A para coleta da corrente e com uma outra ponta de prova, LF-20A, para coleta da tensão. Com ambas as informações sendo recuperadas do osciloscópio através da conexão USB do mesmo, utilizando o software IntuiLink Data Capture for Agilent Osciloscopes v3.90.

Figura 7 - Esquema de ligação dos equipamentos (ARM)

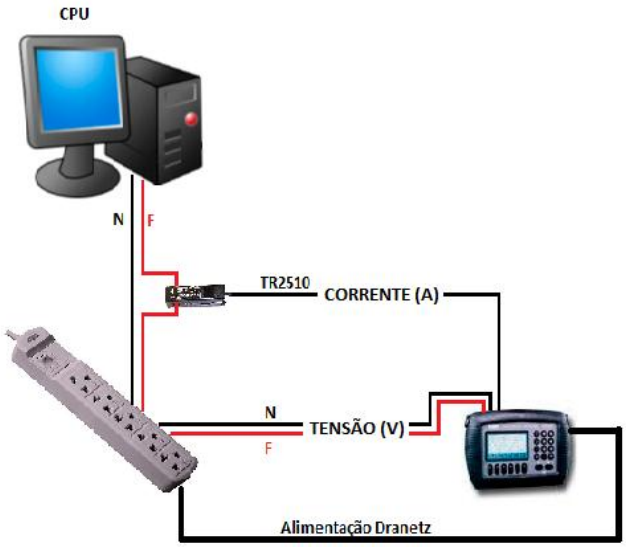


Os testes realizados com a CPU desktop Leader tech foram procedidos da mesma forma que os testes realizados com a BeagleBoard onde inicialmente se adquiriu informações com relação ao consumo do equipamento em estado ocioso, em seguida foram executados os programas repetidas vezes, para se obter um valor médio de consumo de energia durante a execução dos mesmos, e o tempo médio de execução do programa com cada uma das otimizações. Já para a execução do HPL, foram realizadas várias modificações no arquivo HPL.dat, para tornar o

benchmark o mais ajustado possível à arquitetura do equipamento medido, quando se obteve o melhor resultado, foi realizado novamente a execução do algoritmo, e ao mesmo tempo foi mensurado o consumo de energia do equipamento para executar tal programa.

As medições de consumo do CPU foram realizadas através da Plataforma BMI Dranetz PP4300, equipada com uma ponta de prova de corrente TR2510 e duas ponteiras para a medição da tensão que eram ligadas na mesma régua que a CPU e ligadas nos polos positivo e negativo do canal A do Dranetz. Os dados adquiridos pelo Dranetz foram recuperados através do cartão de memória flash, contido no equipamento, e armazenados em um laptop. Para manipular os dados recuperados do Dranetz, utilizou-se o software Dran-View Enterprise 6.11.02, software proprietário desenvolvido pela própria fabricante do aparelho e única forma de visualizar os dados adquiridos pelo mesmo.

Figura 8 - Esquema de ligação dos equipamentos CPU



5.1. Descrição dos Equipamentos

Abaixo, configurações especificas dos equipamentos medidos, apresentando especificações como o processador utilizado, o sistema operacional que foi instalado nas máquinas e o compilador utilizado para compilar os programas executados.

5.1.1. A CPU

A CPU Leader Tech é um computador pessoal desktop, desenvolvido com o objetivo de obter o melhor desempenho possível. A CPU possui internamente ligados à fonte, a placa-mãe, o processador, uma placa de vídeo, um drive de CD, um drive de disquete e um disco-rigido.

A CPU possui um processador Intel Pentium IV, 3.01 GHz com tecnologia de HT (Hyper-Threading), com conjunto de instruções 32 bits, FSB 800MHz. Memória cache L1 de 32 bits de instruções e 32 bits de cache, e cache L2 de 1024KB.

A memória RAM são dois pentes de memória DDR com frequência de 333MHz.

O computador é alimentado por uma fonte iMBP 400W modelo: KYP-280ATXN com uma entrada AC 115/230V 4/2A e frequência entre 50/60Hz.

5.1.2. A BeagleBoard

A Beagleboard é um dispositivo alimentado por USB, de baixo custo, que não necessita de cooler, é uma placa única, com o desempenho e a expansibilidade de um computador, mas sem o volume, a despesa e o ruído causado por máquinas desktop típicas (BEAGLEBOARD, 2009).

A Beagleboard utiliza um processador OMAP3530DCBB72 a 720MHz e vem com um pacote POP de .4mm . POP (pacote em pacote) é uma técnica onde as memórias, NAND e SDRAM, são montadas sobre o OMAP3530.

Quanto à alimentação da placa, ela pode ser feita através de um cabo USB alimentado por uma CPU ou então através de uma fonte. Utilizou-se a alimentação por cabo USB, pois não tinha-se uma fonte para alimentar a placa.

Figura 9 - Especificações de Energia BeagleBoard

Specification	Min	Typ	Max	Unit
Power				
USB Host (Same as the DC supplied by the power plug or USB 5V)	4.8	5	5.2	V
Current (Depends on what the DC source can supply over what the board requires)		Varies		

Fonte: BeagleBoard System Reference Manual.

Tabela 1 - Especificações ARM e x86

Configuração	BeagleBoard Rev3	Pentium IV
Processador	ARM Cortex-A8	Intel Pentium 4 HT 3.0E
Modelo	TI OMAP 3530	Especificação SL7PM
Frequencia de clock	600 MHz	3.014 GHz
processadores	1	1
# core	1	1
Threads/core	1	2
total threads	1	2
RAM	256MB LPDDR RAM e 256 MB NAND Flash	512 MB DDR 333MHz
Cache L1	32 KB	32KB dados + 32KB instruções
Cache L2	256 KB	1024Kb
Cache l3	-	-
TDP maxima (Watts)	2 W	84W
SO	Ubuntu 11.04	Ubuntu 11.04
Kernel	2.6.38-4	2.6.38-8
Compilador	arm-linux-gnueabi-gcc-4.5.2	gcc-4.5.2

5.2. Ambiente de Execução:

Neste capítulo apresenta-se o ambiente de execução do trabalho, especificando o Sistema Operacional utilizado e o motivo, o compilador escolhido para compilar os programas e o software livre utilizado como benchmark.

5.2.1. Sistema Operacional:

O sistema operacional escolhido para a instalação nos equipamentos para a mensuração do desempenho e do consumo foi o Ubuntu 11.04 (Natty Narwhal).

Uma das principais motivações da escolha por este, é que o mesmo possui suporte completo a microarquitetura ARM, desde a sua versão 9.04 (Jaunty Jackalope).

A instalação do sistema operacional Ubuntu 11.04 no cartão de memória SD utilizado na placa BeagleBoard REV C3, foi realizada conforme UbuntuWiki (2011).

A versão do kernel nos equipamentos durante as execução dos testes, eram 2.6.38-4 na placa com processador ARM e 2.6.38-8 no computador com processador x86.

Para compilar os algoritmos utilizou-se por padrão o GCC versão 4.5.2 que está disponível para ambas as arquiteturas de processadores. Para compilar o algoritmo, foram utilizadas algumas das optimizações fornecidas pelo gcc, para este trabalho, optou-se por utilizar as optimizações default (O), O1 e O2.

Conforme Free Software Foundation, Inc. (2011) :

- (default-O) com esta flag, o gcc apenas compila o código, não gerando nenhum tipo de otimização;
- -O1 – com esta flag o gcc leva um pouco mais de tempo para compilar, otimizando um pouco a execução do programa;
- -O2 – com esta flag, o gcc executa quase todas as otimizações suportadas que não envolvem um compromisso entre espaço e velocidade. Utilizando-se –O2, comparado com a flag default, o gcc aumenta tanto o tempo necessário para a compilação do código, quanto o desempenho do código gerado.

Os algoritmos utilizados podem ser vistos no Anexo H e Anexo I, juntamente com a biblioteca criada para cronometrar o tempo de execução dos mesmos no Anexo J.

5.3. Linpack (HPL)

Inicialmente o HPL era um conjunto de rotinas em Fortran usado como benchmark para supercomputadores criados no início dos anos 80. Criado por Jack Dongarra, o teste era simplesmente a resolução de matrizes lineares 100X100. Este teste se tornou inviável assim que a matriz passou a caber na cache de um microprocessador.

Como a primeira implementação do benchmark se tornou inviável, foi desenvolvida uma nova versão onde o tamanho da matriz pode ser definido pelo usuário. Atualmente a implementação portável do Linpack passou a se chamar High Performance Linpack, sendo a principal ferramenta de avaliação de performance em supercomputadores (inclusive clusters).

O Linpack é o principal fator de avaliação para compor a lista dos 500 computadores mais rápidos do mundo, o Top500.

O desempenho do HPL é fortemente dependente da biblioteca BLAS, existindo várias versões que podem ser utilizadas.
A fatoração do benchmark HPL requer (2/3)*N^3 operações e cada duas soluções triangulares totalizam N^2 operações cada, todas de ponto flutuante. A medida que N aumenta, a fase de fatoração domina o tempo de computação.
Os dados relacionados com a i-ésima iteração do algoritmo são apresentados na Figura 4, D é o i-ésimo bloco na diagonal principal, L,U e T são as partes atuais da matriz inferior, superior e restante.
O valor de N é determinado pela quantidade máxima de memória que cada nó computacional pode destinar para o processo HPL antes que comecem a acontecer operações de swap para o disco. Isso significa que o sistema operacional precisa ser minimizado para permitir mais espaço para a execução do HPL. Assim sendo, a imagem do sistema operacional para rodar os testes HPL não deve ser a mesma que a da fase de produção do cluster.
Assumindo que o tamanho da matriz A é 8*N bytes, então, se em cada nó computacional deixarmos mais ou menos 20% da memória para o sistema, N pode ser escolhido com a seguinte fórmula [11]:

$$\mathrm{N} = \sqrt{(0{,}8 \ \mathrm{X} \ \mathrm{NumNos} \ \mathrm{X} \ \mathrm{MemNo})/8} \quad [1]$$

Onde, Num Nos é o número total de nós e Mem No é a capacidade de memória de cada nó do cluster.
Se N for muito alto, então o sistema começa a paginar e o desempenho diminui sensivelmente. Se N for muito baixo, mais computação poderia ter sido feita em cada nó.

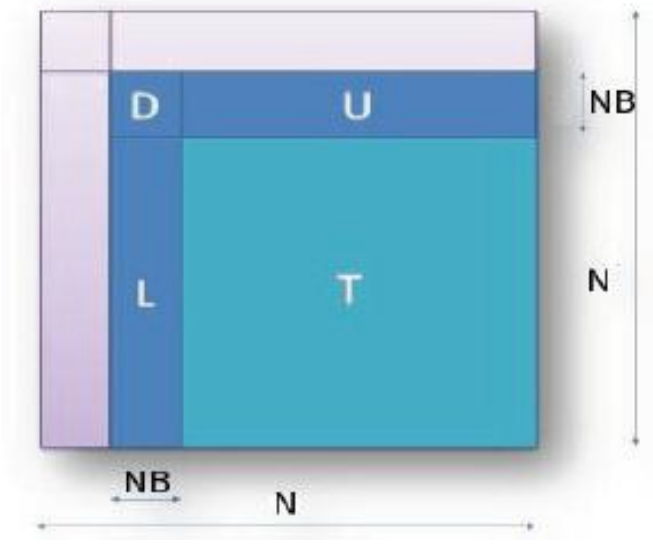


Figura 10 - Fatoração da Matriz.

Para estimar o valor NB, deve-se iniciar com valores de 32 ou 40, e então tentar fazer NB o maior possível. Uma vez encontrado um bom valor para NB, valores de múltiplos pares de NB, expressos pela fórmula 2*i*NB,

devem dar bons resultados também. Como N e NB são relacionados, é desejável que N seja múltiplo de NB para evitar computações residuais. Os valores de P e Q estão relacionados ao número de processos, tal que: P*Q = NumProcessos. A regra de ouro é ter Q > P e possivelmente Q=2*P ou P= sqrt (N) (SILVA, 2009).

6. RESULTADOS

As medições realizadas na etapa inicial do trabalho, tinham por objetivo, adquirir o conhecimento do consumo dos equipamentos em estado ocioso. Os resultados obtidos nestas medições são os seguintes: O processador da arquitetura ARM faz um consumo médio de 0,9431 Wh enquanto o processador x86 faz um consumo de 90,9836 Wh. Com estes valores pode-se observar que o processador arquitetura x86 utilizado, sem realizar nenhuma tarefa consome 96,47 vezes mais energia que o processador da arquitetura ARM.

Na tabela 2, estão descritos os valores médios de consumo para a execução do algoritmo de multiplicação de duas matrizes de ordem 1000(mil) nas duas arquiteturas, executando de forma sequencial e com 2 (duas) threads sem otimização, com otimização –O1 e –O2 respectivamente.

Tabela 2 – Consumo médio na execução do algoritmo

x86 e ARM - Consumo médio na execução (Wh)				
		x86	ARM	x86 / ARM
Sequencial	Otimização O	0,93632 ± 0,0083	0,11130 ± 0,0071	8,41
	Otimização O1	0,69942 ± 0,0056	0,06250 ± 0,0029	11,19
	Otimização O2	0,69595 ± 0,0081	0,06217 ± 0,0045	11,19
Threads	Otimização O	0,87679 ± 0,0397	0,09476 ± 0,0022	9,25
	Otimização O1	0,66870 ± 0,0413	0,05622 ± 0,0022	11,89
	Otimização O2	0,68256 ± 0,0084	0,05498 ± 0,0009	12,41

Na tabela 3 estão descritos os tempos médios de execução do algoritmo, executando de forma sequencial e com 2 (duas) threads, sem otimização, e com otimização –O1 e –O2 respectivamente.

Tabela 3 - Tempo médio de execução do algoritmo

x86 e ARM – Tempo médio de execução (s)				
		x86	ARM	x86 / ARM
Sequencial	Otimização O	16,23 ± 0,083	318,97 ± 4,784	19,65
	Otimização O1	10,46 ± 0,020	182,96 ± 9,961	17,49
	Otimização O2	10,33 ± 0,013	180,79 ± 13,435	17,50
Threads	Otimização O	12,84 ± 0,039	277,49 ± 6,668	21,61
	Otimização O1	9,11 ± 0,010	166,25 ± 6,558	18,25
	Otimização O2	9,10 ± 0,009	160,26 ± 3,136	17,61

Ao analisando as duas tabelas acima, pode-se verificar que executando o algoritmo sequencial sem otimização o processador da arquitetura x86 consumiu 8,41 vezes mais energia do que o processador ARM, mas em contrapartida, levou apenas 5,09% do tempo que o processador ARM levou para executar, ou seja, apesar de consumir 8,41 vezes mais energia, ele foi quase 20 vezes mais rápido.

Pode-se notar também, que a medida que vai se otimizando o algoritmo, a diferença entre o consumo das duas arquiteturas vai aumentando, com o x86 chegando a consumir 12,41 vezes mais energia que a ARM. À medida que a diferença no consumo de energia aumenta a diferença entre os tempos de execução diminui, porém é uma quantia pouco significativa.

A tabela 4 apresenta em percentagem a redução em consumo e em tempo da arquitetura ARM quando o algoritmo foi recompilado utilizando as flags de otimização –O1 e –O2 em sequencial e –O, -O1 e –O2 executando em paralelo, comparando-se com a execução sequencial com a otimização padrão do GCC.

Tabela 4 - % redução de consumo e tempo de execução utilizando otimizações na Arquitetura ARM

			Consumo (Wh)	**Tempo (s)**	**% redução do consumo**	**% redução no tempo**
ARM	Sequencial	O	0,1113018	318,97	100,00%	100,00%
		O1	0,0625016	184,95	43,84%	42,02%
		O2	0,0621746	180,78	44,14%	43,32%
	Thread	O	0,0947602	277,49	14,86%	13,00%
		O1	0,0562187	166,24	49,49%	47,88%
		O2	0,0549800	160,26	50,60%	49,76%

Como se pode ver na tabela acima, ao recompilar o algoritmo sequencial utilizando-se as otimizações fornecidas pelo GCC, obteve-se reduções de 43,84% a 44,14% no consumo de energia e reduções de 42,02% e 43,32% no tempo de execução. Ao se utilizar técnicas de processamento paralelo, com a mesma flag de compilação O, utilizada como padrão do consumo, percebe-se uma redução de quase 15% no consumo em comparação a execução em sequencial, e uma redução de tempo de 13%. Ao utilizarem-se as flags de otimização –O1 e –O2, estes resultados ficam ainda melhores, chegando a reduzir o consumo entre 49,49% e

50,60% e o tempo de execução é reduzido a 47,88% com a flag –O1 e 49,76% com a flag –O2.

A tabela 5 demonstra em percentagem a redução em consumo e em tempo da arquitetura x86 quando o algoritmo foi recompilado utilizando as flags de otimização –O1 e –O2 em sequencial e –O, -O1 e –O2 executando em paralelo, comparando-se com a execução sequencial com a otimização padrão do GCC.

Tabela 5 - % redução de consumo e tempo de execução utilizando otimizações na Arquitetura x86

			Consumo (Wh)	**Tempo (s)**	**% redução do consumo**	**% redução no tempo**
x86	Sequencial	O	0,9363222	16,23	100,00%	100,00%
		O1	0,6994215	10,46	25,30%	35,55%
		O2	0,6959479	10,32	25,67%	36,41%
	Thread	O	0,8767932	12,84	6,36%	20,89%
		O1	0,6687007	9,11	28,58%	43,87%
		O2	0,6825618	9,10	27,10%	43,93%

Pode-se observar na tabela 5, que as otimizações realizadas pelo GCC na arquitetura x86 não alcança resultados tão significativos quanto alcançou na arquitetura arm. Mas ainda assim, é importante destacar os resultados obtidos com esta melhoria. Ao recompilar o algoritmo de execução sequencial com as flags –O1 e –O2, obteve-se uma redução de 25,3% a 25,67% no consumo e de 35,55% e 36,41% no tempo de execução quando comparados com a execução sequencial com otimização –O.

Ao utilizar-se técnicas de processamento paralelo, a flag –O do algoritmo em thread reduz o consumo em 6,36% e o tempo de execução em 20,89% em comparação a execução sequencial com a mesma flag. Já as execuções com as otimização –O1 e –O2 reduziram o consumo em 28,58% e 27,10% e o tempo de execução em 43,87% e 43,93% respectivamente.

Comparando as tabelas 4 e 5, pode-se observar que as reduções de tempo de execução obtidas ao se utilizar as flags de otimização e técnicas de processamento paralelo foram muito parecidas, alcançando quase 50% de melhora na arquitetura ARM e cerca de 44% na arquitetura x86. Já ao comparar a redução no consumo de energia, percebe-se que as arquiteturas apresentaram resultados

bem diferentes. Enquanto na arquitetura ARM, o consumo de energia reduziu na mesma proporção que o tempo de execução, na arquitetura x86, esse redução foi muito menor, não passando de 28,58% de redução no consumo enquanto a redução no tempo de execução chegou a quase 44%.

Na tabela 6 são apresentados os melhores resultados obtidos na execução do Benchmark HPL, presente no pacote do Linpack, a média do consumo durante a execução do mesmo, e o resultado da divisão entre o resultado obtido no benchmark e o consumo médio do aparelho.

Tabela 6 - Desempenho / Watt

Desempenho / Watt			
	x86	ARM	x86 / ARM
Linpack (GFLOPS)	3,986	0,02356	169,185059
Consumo (Wh)	158,38	1,193	132,757754
MFLOPS / W	25,16732	19,7485331	1,27438929

Na tabela acima é possível verificar as diferenças entre as duas arquiteturas durante a execução do benchmark Linpack. Pode-se observar nesta tabela, que o resultado da execução do teste HPL, é quase 170 vezes melhor na arquitetura x86 do que na ARM, o consumo de energia é 132,75 vezes maior, mas no momento em que é verificado o desempenho por Watt destas máquinas, esta diferença fica em apenas 1,27 vezes maior, um valor realmente pequeno tendo em vista as diferenças no resultado do teste e no consumo de energia.

O cálculo realizado na tabela acima para verificar o desempenho por watt foi realizado da mesma forma que é feito na lista top500 onde eles utilizam o melhor resultado obtido no teste HPL e divide-se o mesmo pelo consumo. Na lista top 500 pode-se verificar isso dividindo o valor que consta na coluna Rmax pelo valor da coluna Power e o resultado obtido se encontra na coluna Mflops/Watt.

7. CONSIDERAÇÕES FINAIS

A hipótese que deu início ao trabalho, tinha como tema de pesquisa, averiguar se uma máquina com processador ARM seria capaz de substituir máquinas desktop utilizadas pela maioria dos usuários.

Para a realização do trabalho, foi necessário fazer um estudo mais detalhado a respeito dos sistemas operacionais, a arquitetura dos processadores, conceitos sobre eletricidade e benchmarks.

Com base na hipótese, foram escolhidos alguns tipos de benchmarks a serem executados em uma máquina desktop e em uma máquina com processador ARM com a finalidade de conseguir comparar as duas.

Com relação a desempenho, as duas arquiteturas ainda apresentam uma diferença muito grande em favor da arquitetura x86, sendo entre 17,5 e 21,6 vezes maior nesta arquitetura, por esta ser desenvolvida especialmente para apresentar alto desempenho sem ter em vista a economia de energia. Já o processador da arquitetura ARM, apresentou um consumo de energia realmente reduzido.

Pode-se concluir então, que se o usuário busca um equipamento para fazer uma grande quantidade de processamento, onde é necessária uma resposta imediata do processador e com a execução mais rápida possível, é aconselhável que se continue utilizando os processadores da arquitetura x86, pois se encaixam melhor às necessidades do usuário.

Já ,se o objetivo de uso do equipamento é para utilizá-lo como um servidor de arquivos, ou para fazer downloads, em suma, tarefas que não exijam grande poder de processamento, é recomendável que o usuário busque por tecnologias alternativas, visto que com isso, a redução de consumo de energia pode chegar a quase 100 vezes fazendo o uso de um processador da arquitetura ARM.

8. TRABALHOS FUTUROS

- Realizar testes com otimizações O3 e Os nos mesmos equipamentos para verificar qual das otimizações proporciona um melhor desempenho, com o menor consumo de energia.

- Realizar testes com um número maior de equipamentos de ambas as arquiteturas, visando obter um valor de referência e buscar propor um selo de qualidade de consumo /eficiência no uso de energia baseado no desempenho de processamento dividido pela quantidade de watts consumidos.

- Realizar testes de consumo em diversos computadores, compilando diversos algoritmos com todas as flags de otimização proporcionadas pelos compiladores e verificar o consumo de energia realizado em cada compilação, com a ideia de buscar definir a flag que proporciona o maior ganho de desempenho e a melhor redução no consumo de energia.

- Baseado na tabela 4, criar uma otimização para o GCC que favoreça principalmente a redução no consumo de energia do equipamento.

9. REFERÊNCIAS BIBLIOGRÁFICAS

BEAGLEBOARD. BeagleBoard System Reference Manual Rev C3, 2009. Disponível em <http://beagle.s3.amazonaws.com/design/Beagle_SRM_C3_0_0.pdf> Acessado em nov. 2011.

CARTER, Nicholas. **Arquitetura de Computadores**. São Paulo: Bookman, 2003.

DAS, Lyla B. **The x86 Microprocessors**: Architecture, Programming and Interfacing (8086 to Pentium). India: Pearson Education, 2010.

FOWLER, R. J. **Eletricidade: princípios e aplicações**. 2ª edição. São Paulo: Makron Books, 1992.

FREE SOFTWARE FOUNDATION, INC. **GCC Development Mission Statement.** Disponível em <http://gcc.gnu.org/gccmission.html> Acessado em nov. 2011.

GOMES, P. H.; LEITE, T. S.; CAETANO, U. I. **A Arquitetura ARM**. Disponível em <http://www.ic.unicamp.br/~rodolfo/Cursos/mc722/2s2005/Trabalho/g20-arm.pdf> Acessado em mar. 2011.

GUSSOW, Milton. **Eletricidade básica**. 2ª edição. São Paulo: Makron Books, 1996.

HEWITT, P. G. **Física Conceitual.** 9ª Edição. Porto Alegre: Bookman, 2002.

IDA, Cesar Ossamu **Benchmarks** Disponível em: <http://www.inf.ufrgs.br/gppd/disc/cmp134/trabs/T1/001/benchmarks/Benchmarks_p2.htm> Acessado em nov. 2011.

INTEL. **Introducing the 45 nm Next-Generation Intel ® CORE ™ microarchitecture. White Paper**. Disponível em <http://www.intel.com/technology/architecture-silicon/intel64/45nm-core2_whitepaper.pdf> Acessado em mar. 2011.

LILJA, David. J. **Measuring computer performance:** A practitioner´s guide. Cambridge (UK): Cambridge University Press, 2000.

MANO, M. Morris. **Computer system architecture**. 3ª edição. Englewood Cliffs (EUA): Prentice-Hall,1993.

MALUTA, T. **Comparações de Performance**: ARM vs. x86 Em Algoritmos Simples de Ordenação. Wordpress, 2008. Disponível em <http://escovandobits.files.wordpress.com/2008/04/arm_vs_x86.pdf> Acesso em mar. 2011.

MARINHA DOS ESTADOS UNIDOS. **Curso de eletricidade Básica**. 1ª edição. Rio de Janeiro: Hemus, 1998.

MORIMOTO, C. E. Chips ARM: Entendendo as diferenças entra o Hummingbird, Snapdragon, OMAP e Tegra. **Hardware.** Disponível em <http://www.hardware.com.br/artigos/arm-entendendo-diferencas/> Acesso em mar. 2011.

MURDOCCA, Miles J.; HEURING, Vincent P. **Introdução à Arquitetura de Computadores**. Rio de Janeiro: Campus, 2000.

OBAIDAT, Mohammad S.; BOUDRIGA, Noureddine A. **Fundamentals of performance evaluation of computer and telecommunication systems.** New Jersey: John Wiley & Sons, 2010.

O'MALLEY, J. **Análise de Circuitos**. 2ª edição. São Paulo: Makron Books, 1993.

PRADO, Claudio. L; SILVA, João M. A. **Aplicação de cluster Beowulf em instituições de ensino.** 2010. 119 f. Monografia (Graduação em Tecnologia em Informática, ênfase em Redes de Computadores) – Faculdade de Tecnologia de Guaretingueta, Guaretingueta, 2010.

PRADO, S. **A onipresente arquitetura ARM**. Disponível em <http://www.sergioprado.org/2011/02/24/a-onipresente-arquitetura-arm/> Acessado em mar. 2011.

RAMOS, Nelson A. **Avaliação e comparação de desempenho de computadores: metodologia e estudo de caso.** 2008. 60 f. Monografia (Graduação em Engenharia da Computação) – Universidade Federal de Pernambuco, Recife, 2008.

RAUBER, Thomas; RÜNGER, Gudula. **Parallel programming for multicore and cluster systems.** 1ª edição. New York: Springer, 2010.

RODITI, I. **Dicionário Houaiss de física.** 1ª edição. Rio de Janeiro : Objetiva, 2005.

ROQUE, Tânia. **Como funciona um osciloscópio.** 2007. Disponível em <http://www.notapositiva.com/pt/trbestbs/fisica/11osciloscopio.htm> Acessado em nov. 2011.

SILVA, Vinícius; BENTES, Cristiana; GUEDES, Sérgio; SILVA, Gabriel P.; **Arquitetura e avaliação do cluster de alto desempenho Netuno.** In: WSCAD-SSC 2009 - X Simpósio em Sistemas Computacionais, 2009, São Paulo. P. 52-59.

STALLINGS, William. **Arquitetura e organização de computadores**: Projeto para o Desempenho. São Paulo: ABDR, 2002.

TORRES. G; LIMA. C. **Por dentro da arquitetura do Pentium 4**. Disponível em <http://www.clubedohardware.com.br/artigos/Por-Dentro-da-Arquitetura-do-Pentium-4/1201/4> Acessado em nov. 2011.

TSUKAMOTO, Steven K.; **Arquitetura dos supercomputadores – TOP500 -.** 2010. 16 f. Monografia (Graduação em Ciências da Computação) – Universidade de São Paulo, São Paulo, 2010.

UBUNTUWIKI. **Ubuntu 11.04 (Natty Narwhal)**. Disponível em <http://cdimage.ubuntu.com/releases/11.04/release/> Acessado em nov. 2011.

ANEXOS

Anexo A Configuração Osciloscópio

A configuração do osciloscópio foi realizada conforme manual do usuário disponível em < http://cp.literature.agilent.com/litweb/pdf/54130-97014.pdf >.

Segue abaixo, manual de configuração do osciloscópio utilizado no trabalho:

Ligar o Osciloscópio;

Colocar a ponta de prova de corrente (Agilent 1146A AC/DC current probe) no canal 1;

Ligar a Ponta de Prova em 100mV/A;

Apertar o botão 1;

apertar em acoplamento;

Selecionar o tipo de acoplamento DC;

E apertar botão push to select;

Apertar em Ponta Prova;

Selecionar Ponta Prova 1X;

E apertar botão push to select;

Apertar no botão ½ e com uma flecha para baixo;

Apertar em unidade;

Selecionar a unidade A (corrente);

Apertar o botão push to select;

Apertar o botão measure;

Apertar o botão fonte até que o valor abaixo de fonte seja CH1;

Apertar o botão limpar;

Apertar o botão tensão;

Selecionar a opção Vrms e apertar o botão push to select;

Inserir a ponta de prova de tensão (Ponta de Prova de Osciloscópio LF-20A) no canal 2;

Apertar o botão 2;

apertar em acoplamento;

Selecionar o tipo de acoplamento DC;

E apertar botão push to select;

Selecionar Ponta Prova 1X;

E apertar botão push to select;

Apertar no botão ½ e com uma flecha para baixo;

Apertar em unidade;

Selecionar a unidade V (tensão);

Apertar o botão push to select;

Apertar o botão measure;

Apertar o botão fonte até que o valor abaixo de fonte seja CH2;

Apertar o botão limpar;

Apertar o botão tensão;

Selecionar a opção Vrms e apertar o botão push to select;

Pronto o osciloscópio está pronto para exibir os valores de Corrente e Tensão a partir dos canais 1 e 2 respectivamente.

Anexo B Calibrando a Ponta de Prova 1146A

Configuração Da ponta de prova 1146A para o osciloscópio Agilent DSO1002A:

Aparelho Utilizado para a Configuração:

Fonte ICEL PS-1500.

Ligar os equipamentos.

Configurar Ponta de prova de corrente Agilent 1146A AC/DC current probe conforme especificado no Tutorial de configuração do Osciloscópio;

Após isto, Com uma ponteira com ponta jacaré, dar 10 voltas na ponta de prova.

Ligar a ponteira na fonte ICEL PS-1500 no polo positivo, e com a ponta jacaré ligar esta em outra ponteira jacaré que será ligada no polo negativo da fonte.

Ligar a ponta de prova em 100mV/A;

Ligar a fonte;

Girar o regulador de tensão da fonte até que no indicador da corrente se acenda uma luz vermelha;

Girar o regulador de corrente até 0,40 A.

Desta forma, a fonte ira gerar uma corrente de 0,40 A ou 400mA;

A partir de agora, o único equipamento em que será alterado alguma coisa é na ponta de prova de corrente do osciloscópio;

Provavelmente, o valor que ira ser exibido na tela do osciloscópio em mA, será diferente do valor que esta sendo gerado pela fonte;

Se o valor apresentado na tela do osciloscópio for maior que o valor gerado pela fonte, você deve girar o regulador da ponta de prova no sentido anti-horário. De outra forma, se o valor apresentado no visor do osciloscópio for menor que o gerado pela fonte, você deve girar o regulador no sentido horário.

Quando os valores apresentados pelo osciloscópio e pela fonte forem idênticos, considera-se que a ponta de prova de corrente do osciloscópio está configurada corretamente, e pode-se começar a fazer as medições;

Anexo C Utilização do Software IntuiLink Data Capture

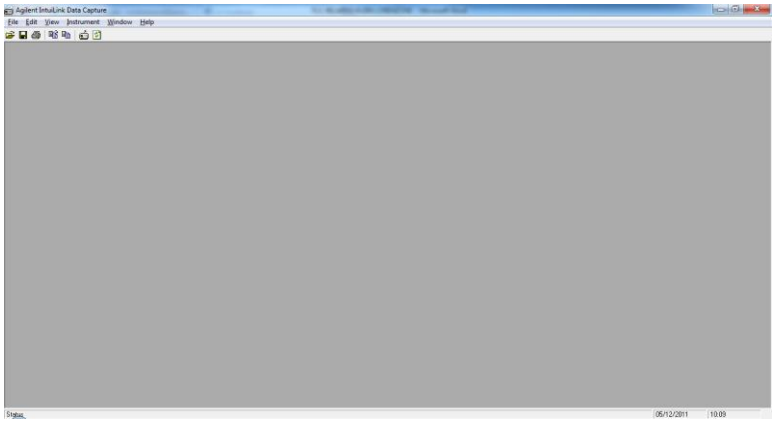
O software IntuiLink Data Capture 3.9.0 esta disponível para download em <http://download.home.agilent.com/1A/AgilentDataCapture0390.exe?authparam=1323051272_6e02175f63a1e33b0a8ecc800dfb6347>.

Segue descrição passo a passo para utilização do software IntuiLink Data Capture, utilizado para recuperar os dados capturados pelo osciloscópio.

Conecte o cabo USB no osciloscópio e no computador com o Software IntuiLink Data Capture for Agilent Oscilloscopes v3.90 já instalado, assim como a biblioteca Agilent I/O Libraries Suite 16.1 que o software necessita.

Abra o software IntuiLink Data Capture, clique no menu Instrument;

Figura 11 - Tela principal do IntuiLink Data Capture 3.90



Fonte: Print da tela principal do software.

Selecione a opção Agilent 1000 Series;

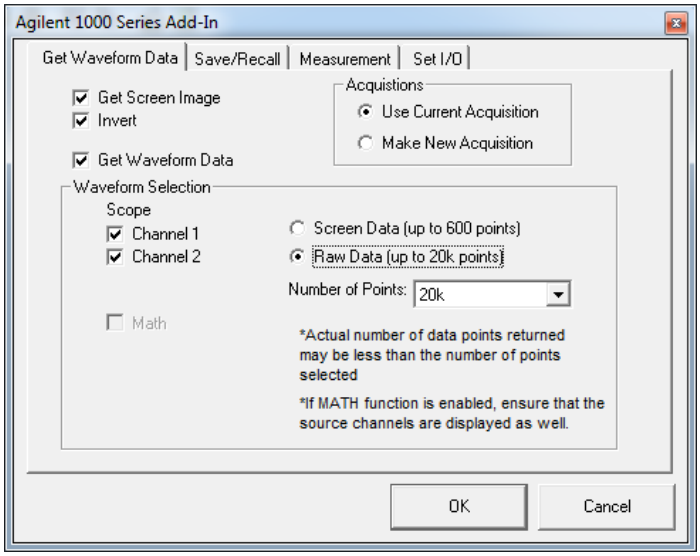
Na nova janela, na aba Set I/O Clique no botão Find Intrument;

O programa irá reconhecer o equipamento que esta conectado via usb selecione a opção e de dois cliques nela, o equipamento ira aparecer no campo Indintified instruments on my computer, selecione ele e clique em ok;

Desta forma você já selecionou o equipamento do qual ira recuperar os dados;

Va para a aba Get Waveform Data e certifique-se de fazer as alterações para que fique conforme a imagem abaixo.

Figura 12 - Tela de configuração para recuperar os dados do osciloscópio



Fonte: Print do software em execução.

Ao concluir a configuração nesta aba, aperte o botão run/stop no osciloscópio antes de iniciar a execução do algoritmo e, ao termino do mesmo, aperte novamente o botão para pausar a coleta de dados. Certifique-se de que todo o intervalo de coleta dos dados esteja visível na tela do osciloscópio, pois devido ao tamanho da memória do mesmo as informações do inicio da coleta serão sobrescrevidas caso o gráfico ultrapasse o fim da tela. Então basta clicar em Ok ainda na janela de configuração para que o software recupere os dados fornecidos pelo osciloscópio;

Esta configuração deverá ser executada toda vez que o software for iniciado. Após a primeira configuração, basta clicar no botão Get Data, identificado pelo desenho de um osciloscópio e localizado na barra de ferramentas da aplicação, para recuperar os dados novamente.

Para reiniciar a coleta de dados no osciloscópio após a recuperação dos mesmos, Aperte no osciloscópio o botão force (local) e então inicie a coleta de dados novamente apertando o botão run/stop.

Após recuperar os dados gerados pelo osciloscópio, você terá abertos no IntuiLink, 3 janelas, uma com os waveforms da tensão, uma com os waveforms da corrente e uma imagem, que apresenta a você a tela do osciloscópio no momento da recuperação das informações. As waveforms foram salvas no computador em formato csv, após isto foram abertos em excel e manipulados para então serem utilizados no trabalho.

Anexo D Boot BeagleBoard pela Serial

Este anexo apresenta a sequencia de comandos utilizados para fazer o boot na BeagleBoard através de um computador conectado a mesma pela porta serial.

$ sudo apt-get install gcc-4.5-arm-linux-gnueabi

$ arm-linux-gnueabi-gcc mult_sequencial.c

$ sudo apt-get install minicom
$ sudo minicom -s
--- configuração da porta serial
---------- A - Dispositivo Serial ---> /dev/ttyS0
---------- F - Controle de Fluxo por Hardware ---> NÃ£o
--- sair do minicon

$ sudo minicom
OMAP3 beagleborad.org# mmcinit
OMAP3 beagleborad.org# bootd

omap login: ubuntu
password: temppwd

Anexo E Configuração Dranetz Power Platform 4300

Abaixo apresentamos três manuais sobre o aparelho utilizado nas medições de energia, sendo os manuais referentes à configuração inicial, configuração para a medição de energia e procedimentos para salvar os dados em cartão de memória, transferência dos arquivos para o computador e operações com o software Dran View.

O manual será composto pelo passo à passo formado por imagens, com observações caso necessário.

Configuração inicial

A configuração do equipamento Dranetz Power Platform 4300 foi realizada conforme o manual de usuário disponível em <http://dranetz.com/pdf/powerplatform4300-multi-daq-taskcard-operatorsmanual>.

Figura 13 - Ligando o aparelho.

Fonte: Laboratório do DCEEng.

Figura 14 – Inicialização do Dranetz.

Fonte: Laboratório do DCEEng.

Após inicializar, o aparelho questiona se desejamos selecionar sondagem de corrente? Basta pressionar o botão correspondente a opção YES.

Figura 15 - Selecionando a sondagem de corrente.

Fonte: Laboratório do DCEEng.

Na tela seguinte, devemos selecionar o tipo de sonda utilizada (TR2510), em nosso caso, utilizamos apenas o canal A ignorando os demais canais. Neste caso já está correto o modelo da sonda, basta pressionar EXIT.

Figura 16 - Selecionando o tipo de sonda utilizada (TR2510).

Fonte: Laboratório do DCEEng.

Na tela principal do aparelho, pressionamos o botão SETUP para acessar as configurações necessárias antes das medições.

Figura 17 – Tela principal do aparelho (acessando a tela de configurações).

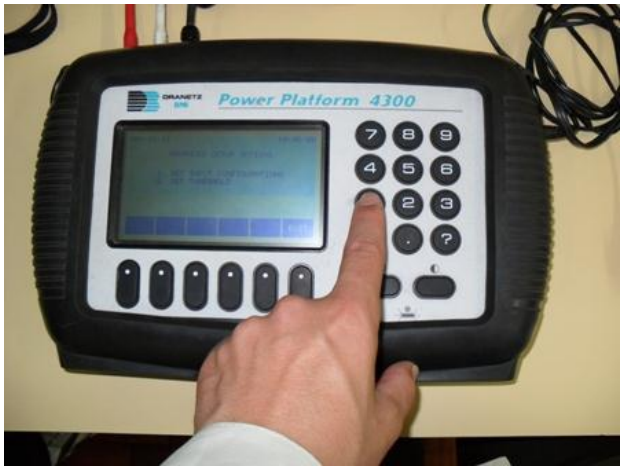
Fonte: Laboratório do DCEEng.

A sequência de imagens abaixo ilustra os passos para a configuração da escala de fatores.

Figura 18 - Acessando a opção 5: ADVANCED SETUP OPTIONS.

Fonte: Laboratório do DCEEng.

Figura 19 – Acessando a opção 1: SET INPUT CONFIGURATIONS



Fonte: Laboratório do DCEEng.

Figura 20 – Acessando a opção 3: SET SCALE FACTORS.

Fonte: Laboratório do DCEEng.

Ajustamos cada canal para (utilizando as opções “NEXT CHAN” e “ENTER SCALE” e pressionando o teclado numérico para a entrada dos números) que fique conforme a imagem abaixo, lembrando que utilizaremos apenas o canal A para realizar as medições.

Figura 21 – Ajuste do fator de escala dos canais.

Fonte: Laboratório do DCEEng.

Após pressionamos duas vezes o botão EXIT para retornar ao menu principal das configurações.

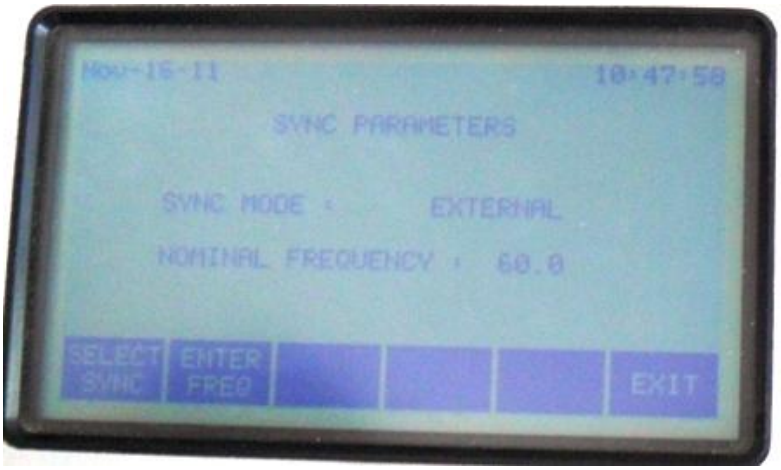
No menu principal de configurações, apresentamos os passos para configurar a frequência de operação para 60Hz.

Figura 22 - Acessando a opção 4: SET SYNC PARAMETERS

Fonte: Laboratório do DCEEng.

A opção “SYNC MODE” deverá ficar como “External” e a opção “NOMINAL FREQUENCY” deverá ficar com o valor “60.0” conforme figura abaixo.

Figura 23 – Ajustando a frequência da rede elétrica.



Fonte: Laboratório do DCEEng.

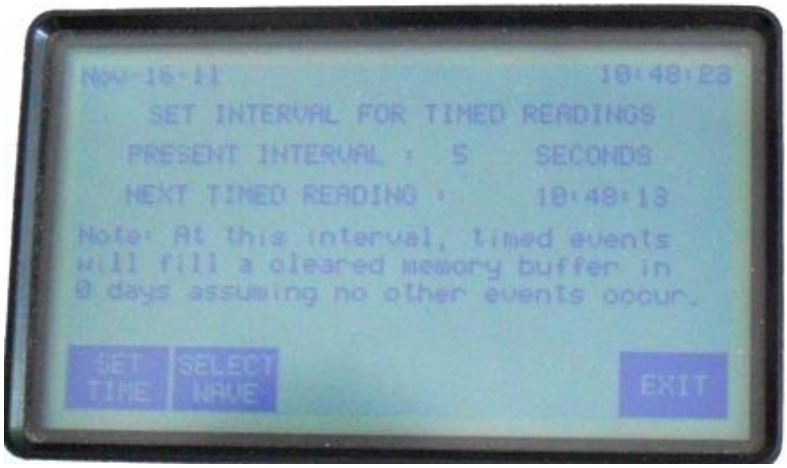
Após pressionamos duas vezes o botão EXIT para retornar ao menu principal das configurações.

O último ajuste necessário está no intervalo em segundos de cada medição, sendo demonstrado na sequencia de figuras abaixo.

Figura 24 - Acessando a opção 5: SET INTERVAL FOR TIMED READINGS.

Fonte: Laboratório do DCEEng.

Figura 25 - Ajustando o intervalo em 5 segundos de medição dos dados.



Fonte: Laboratório do DCEEng.

Após pressionamos EXIT 3 vezes e voltamos a tela principal do Drantz, concluindo assim a configuração inicial do mesmo.

Realizando as medições de energia

A sequência de figuras abaixo ilustra o passo a passo realizado no aparelho em cada medição de energia, salientando que a cada medição, o aparelho era desligado para a remoção do cartão de memória para coleta dos dados.

Figura 26 - Ligando o aparelho.

Fonte: Laboratório do DCEEng.

Figura 27 - Inicialização do Dranetz.

Fonte: Laboratório do DCEEng.

Após inicializar, o aparelho questiona se desejamos selecionar sondagem de corrente? Basta pressionar o botão correspondente a opção YES.

Figura 28 – Confirmando a utilização de sondagem de corrente.

Fonte: Laboratório do DCEEng.

Na tela seguinte, devemos selecionar o tipo de sonda utilizada (TR2510), em nosso caso, utilizamos apenas o canal A ignorando os demais canais. Neste caso já está correto o modelo da sonda, basta pressionar EXIT.

Figura 29 – Definindo o tipo da sonda utilizada.

Fonte: Laboratório do DCEEng.

Agora vamos desativar os canais (B, C e D) que não iremos utilizar pressionando o botão SELECT CHANNEL

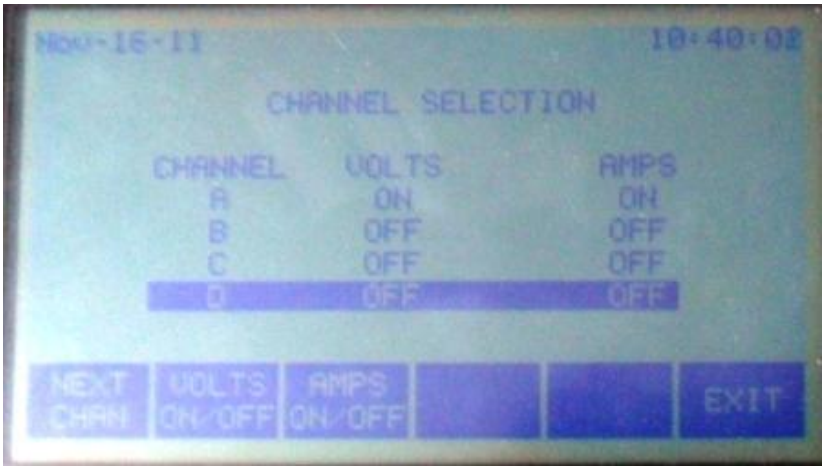
Figura 30 – Acessando menu para desabilitar canais desnecessários.

Fonte: Laboratório do DCEEng.

Para desativar um canal, basta apertar o botão “NEXT CHANEL” para trocar de canal (A, B, C ou D) e após pressionar os botões “VOLTS ON/OFF” e “AMPS ON/OFF” para desativar as medições nos canais desejados (B, C e D)

Ao finalizar de desativar a medição nos canais B, C e D o resultado deverá ser conforme a imagem na sequencia. Após, basta pressionar o botão EXIT e retornar a tela principal.

Figura 31 – Canais B, C e D desativados.



Fonte: Laboratório do DCEEng.

Para acompanhar melhor a medição de energia, na tela principal pressionamos o botão “METER MODE”.

Figura 32 – Alterando a visualização da medição de energia.

Fonte: Laboratório do DCEEng.

Após pressionamos NEXT METER para a exibição da medição desejada.

Figura 33 - Alterando para a próxima visualização da medição de energia.

Fonte: Laboratório do DCEEng.

Exibição da tela de medição desejada. Para sair da tela, basta pressionar EXIT.

Figura 34 – Tela utilizada para acompanhar as medições de energia.

Fonte: Laboratório do DCEEng.

Salvando os dados e exportando para arquivo TXT

Com o equipamento já ligado e em funcionamento, na tela principal, basta acessar a opção SETUP para acessar as funções do cartão de memória.

Figura 35 – Acessando o menu de configurações.

Fonte: Laboratório do DCEEng.

Figura 36 - Acessando a opção 4: MEMORY FUNCTIONS.

Fonte: Laboratório do DCEEng.

Figura 37 - Acessando a opção 1: MEMORY CARD FUNCTIONS.

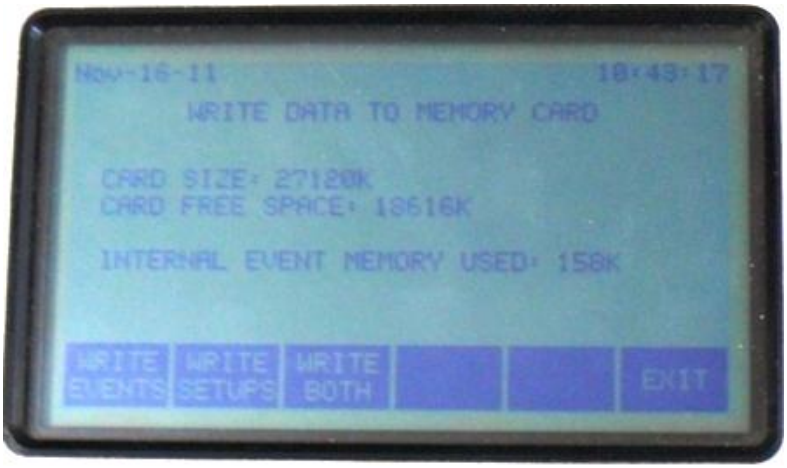
Fonte: Laboratório do DCEEng.

Figura 38 - Acessando a opção WRITE CARD para salvar os dados coletados.

Fonte: Laboratório do DCEEng.

Para salvar os dados no cartão, basta pressionar as opções WRITE EVENTS, WRITE SETUPS e WRITE BOTH.

Figura 39 - Salvar os dados coletados no cartão.



Fonte: Laboratório do DCEEng.

Após salvar os dados, basta desligar o aparelho para remover o cartão de memória.

Figura 40 – Removendo o cartão de memória na frontal direita do aparelho.

Fonte: Laboratório do DCEEng.

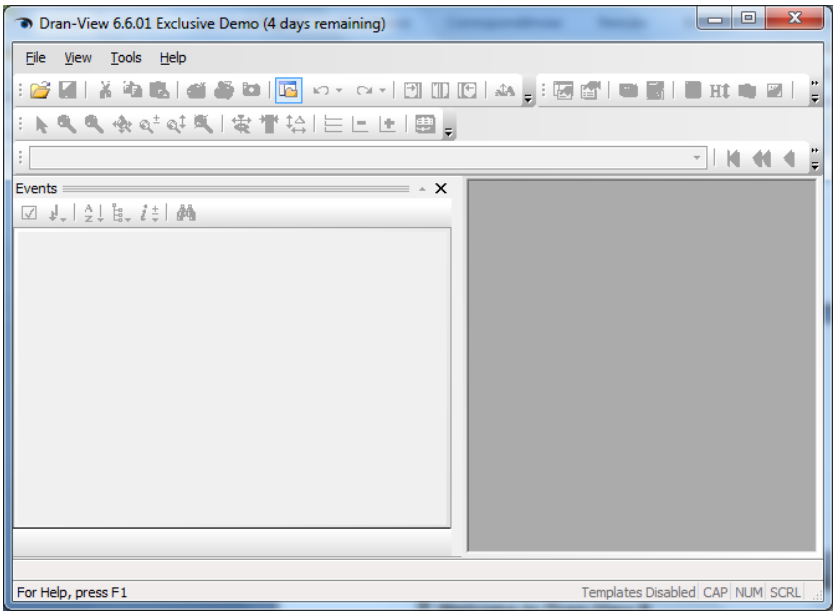
Figura 41 - Salvando os dados para o computador

Fonte: Laboratório do DCEEng.

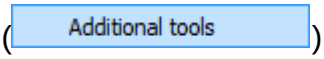
São 3 arquivos (HCI.EVT, HCI.SET, HCI.MDB) que o Dranetz gera, os quais salvamos no computador, sendo utilizado apenas o arquivo HCI.EVT devido ao fato de ser armazendo neste arquivo os últimos 20 minutos de medição, onde a cada medição de 10 minutos, salvava-se os dados no computador, não sendo necessários os demais arquivos.

Para visualizar os dados gerados, utilizamos o software Dran-View 6.0 Entrerprise (trial). A versão do programa utilizada foi a “Enterprise” que está disponível para download em <http://dranetz.com/dran-view-6.11.02-trial-zip >.

Figura 42 - Tela principal do Dran-View.



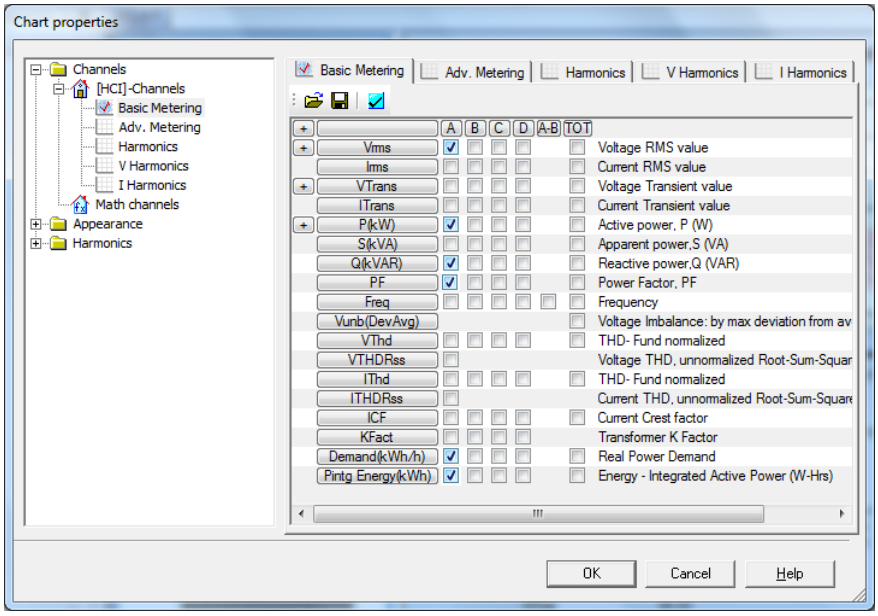
Fonte: print do software em execução.

Para efetuarmos a exportação dos arquivos, precisamos habilitar uma barra de ferramentas (Adittional Tools), habilitamos esta barra ao clicar com o botão direito do mouse sobre alguma das barras já existentes, selecionando a opção Additional Tools (

), a qual disponibiliza o botão properties ( ).

Após, basta abrir os arquivos salvos, clicar no botão “properties” acima citado e selecionar o que desejamos exportar, conforme pode ser visualizado na imagem abaixo.

Figura 43 - Selecionando as informações desejadas.



Fonte: print do software em execução.

Na sequencia, basta ir no menu “FILE” opção “Save AS” e salvar o arquivo no local desejado com o formato TXT (exemplo de um arquivo gerado no anexo X).

Após o arquivo TXT salvo, efetuamos a importação do mesmo com o Microsoft Excel, convertendo o mesmo em tabela e realizando os cálculos apresentados no estudo de caso.

Anexo F HPL.dat

Abaixo o arquivo HPL.dat utilizado para configurar a execução do Linpack HPL.

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
5800         Ns
1            # of NBs
592          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
1            Ps
1            Qs
16.0         threshold
1            # of panel fact
0 1 2        PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
2 4          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0 1 2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

Anexo G Melhor resultado HPL no processador x86

Este arquivo foi gerado pelo HPL, demonstrando as configurações recebidas através do arquivo HPL.dat, e o resultado alcançado pela execução do benchmark.

```
================================================================================
HPLinpack 2.0  --  High-Performance Linpack benchmark  --   September 10, 2008
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================
An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.
The following parameter values will be used:
N         :    5800
NB        :     592
PMAP : Row-major process mapping
P         :       1
Q         :       1
PFACT     :    Left
NBMIN     :       2
NDIV      :       2
RFACT     :    Left
BCAS :    1ring
DEPTH     :       0
SWAP : Mix (threshold = 64)
L1        : transposed form
U         : transposed form
EQUIL     : yes
ALIGN     : 8 double precision words
--------------------------------------------------------------------------------
- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00L2L2        5800   592     1     1              32.65              3.986e+00
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0054613 ..... PASSED
================================================================================
Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------
End of Tests.
================================================================================
```

Anexo H Algoritmo mult_sequencial.c

Algoritmo de multiplicação de matrizes com processamento sequencial.

```
#include <stdio.h>
#include "tempo.h"

#define L 1000
#define C 1000

int m1[C][L], m2[C][L], m3[C][L];

void inicializa_matriz();
void mostra_matriz();
void multiplica();
void mostra_resultado();

int main(int argc, char *argv[])
{

        inicializa_matriz();

//      mostra_matriz();

        tempo1();

        multiplica();

        tempo2();

//      mostra_resultado();

        tempoFinal("mili segundos", argv[0], MSGLOG);

}


void multiplica()

{
        int i, j , v;

      printf("\n  multiplicando.... \n");
        for (i = 0; i < L; i++)
                for (j = 0; j < C; j++)
                        for (v = 0; v < C; v++)
                                m3[i][j] = m3[i][j] + m1[i][v] * m2[v][j];

}
```

```
void mostra_matriz()
{
	int z, v;
	printf("\n  M A T R I Z \n");

	printf("\n  M.1 \n");
	for (z=0; z < L; z ++){
		for (v=0; v < C; v++) {
		printf("  %3d", m1[z][v]);
		}
	  printf("\n");
	  }
	  printf("\n\n\n");

	printf("\n  M.2 \n");
	  for (z=0; z < L; z ++){
		for (v=0; v < C; v++) {
		printf("  %3d", m2[z][v]);
		}
	  printf("\n");
	  }
}
void mostra_resultado()
{
	  int z, v;
	printf("\n  M A T R I Z  R E S U L T A D O \n");

	for (z=0; z < L; z ++){
		for (v=0; v < C; v++) {
		printf(" %4d", m3[z][v]);
		}
	  printf("\n");
	  }
	  printf("\n\n\n");

}

void inicializa_matriz()
{
	int z, v;
	printf("\n inicializando matrizes ...\n");
	for (z=0; z < L; z ++)
		for (v=0; v < C; v++){
			m1[z][v] = z+1;
			m2[z][v] = v+2;
			m3[z][v] = 0;
		}
}
```

Anexo I Algoritmo mult_thread.c

Algoritmo de multiplicação de matrizes com execução em threads.

```
#include <stdio.h>
#include <pthread.h>
#include "tempo.h"

#define L 1000
#define C 1000

int m1[C][L], m2[C][L], m3[C][L];

void inicializa_matriz();
void mostra_matriz();
void *multiplica(void *arg);
void mostra_resultado();

int main(int argc, char *argv[])
{


        pthread_t t1, t2;
        int x,y;

          inicializa_matriz();

//        mostra_matriz();

          tempo1();

          x = 0;
          pthread_create(&t1, NULL, multiplica, (void *)&x);

          y = C/2;
          pthread_create(&t2, NULL, multiplica, (void *)&y);

          pthread_join(t1, NULL);
          pthread_join(t2, NULL);

          tempo2();

//        mostra_resultado();

          tempoFinal("mili segundos", argv[0], MSGLOG);

          pthread_exit(NULL);
}
```

```
void * multiplica(void *arg)

{       int pos = *((int*)arg);

        int i, j , v;

      printf("\n  multiplicando.... \n");
        for (i = pos ; i < (pos+C/2); i++ )
                for (j = 0; j < C; j++)
                        for (v = 0; v < C; v++)
                                m3[i][j] = m3[i][j] + m1[i][v] * m2[v][j];

        return(NULL);
}

void mostra_matriz()
{
      int z, v;
      printf("\n  M A T R I Z \n");

      printf("\n  M.1 \n");
      for (z=0; z < L; z ++){
                for (v=0; v < C; v++) {
              printf("  %3d", m1[z][v]);
                }
        printf("\n");
        }
        printf("\n\n\n");

      printf("\n  M.2 \n");
        for (z=0; z < L; z ++){
                for (v=0; v < C; v++) {
              printf("  %3d", m2[z][v]);
                }
        printf("\n");
        }
}
void mostra_resultado()
{
        int z, v;
      printf("\n  M A T R I Z  R E S U L T A D O \n");

      for (z=0; z < L; z ++){
                for (v=0; v < C; v++) {
              printf(" %4d", m3[z][v]);
                }
        printf("\n");
        }
        printf("\n\n\n");
```

```
}

void inicializa_matriz()
{
	int z, v;
	printf("\n inicializando matrizes ...\n");
	for (z=0; z < L; z ++)
		for (v=0; v < C; v++){
			m1[z][v] = z+1;
			m2[z][v] = v+2;
			m3[z][v] = 0;
		}
}
```

Anexo J Biblioteca tempo.h

Biblioteca utilizada na execução dos algoritmos para obter o tempo de execução dos mesmos durante os testes.

```
#include <sys/time.h>

struct timeval t1, t2;
struct timezone tz1, tz2;

char MSGLOG[200];  // armazena o log no arquivo

void tempo1() {
   gettimeofday(&t1, &tz1);
}
void tempo2() {
   gettimeofday(&t2, &tz2);
}
void tempoFinal(const char *medida, char *nomeArquivo, char *msglog) {
   char tempo[50];
   long int micro = (((t2.tv_sec*1000000)+t2.tv_usec) -
((t1.tv_sec*1000000)+t1.tv_usec));
    if (strcmp(medida,"")==0)
      sprintf(tempo,"Tempo total= %.2f s\n", micro / 1000000.0);
    if (strcmp(medida,"micro segundos")==0)
      sprintf(tempo,"Tempo total= %ld us\n", micro);
    if (strcmp(medida,"mili segundos")==0)
      sprintf(tempo,"Tempo total= %.2f ms\n", micro / 1000.0);
    if (strcmp(medida,"segundos")==0)
      sprintf(tempo,"Tempo total= %.2f s\n", micro / 1000000.0);
    if (strcmp(medida,"minutos")==0)
      sprintf(tempo,"Tempo total= %.2f min\n", (micro / 1000000.0) / 60);
    if (strcmp(medida,"horas")==0)
      sprintf(tempo,"Tempo total= %.2f horas\n", ((micro / 1000000.0) / 60)) / 60;
    printf("\n...: %s\n",tempo);
}
```